Efficient Stitchable Task Adaptation

Haoyu He Zizheng Pan Jing Liu Jianfei Cai Bohan Zhuang†
ZIP Lab, Monash University, Australia

Abstract

The paradigm of pre-training and fine-tuning has laid the foundation for deploying deep learning models. However, most fine-tuning methods are designed to meet a specific resource budget. Recently, considering diverse deployment scenarios with various resource budgets, SN-Net [51] is introduced to quickly obtain numerous new networks (stitches) from the pre-trained models (anchors) in a model family via model stitching. Although promising, SN-Net confronts new challenges when adapting it to new target domains, including huge memory and storage requirements and a long and sub-optimal multistage adaptation process. In this work, we present a novel framework, ***E****fficient* ***S****titchable* ***T****ask* ***A****daptation (ESTA), to efficiently produce a palette of fine-tuned models that adhere to diverse resource constraints. Specifically, we first tailor parameter-efficient fine-tuning to share low-rank updates among the stitches while maintaining independent bias terms. In this way, we largely reduce fine-tuning memory burdens and mitigate the interference among stitches that arises in task adaptation. Furthermore, we streamline a simple yet effective one-stage deployment pipeline, which estimates the important stitches to deploy with training-time gradient statistics. By assigning higher sampling probabilities to important stitches, we also get a boosted Pareto frontier. Extensive experiments on 25 downstream visual recognition tasks demonstrate that our ESTA is capable of generating stitches with smooth accuracy-efficiency trade-offs and surpasses the direct SN-Net adaptation by remarkable margins with significantly lower training time and fewer trainable parameters. Furthermore, we demonstrate the flexibility and scalability of our ESTA framework by stitching LLMs from LLaMA family, obtaining chatbot stitches of assorted sizes*[1].

1. Introduction

The paradigm of pre-training and fine-tuning has underpinned modern applications in both vision and language. With off-the-shelf models pre-trained on large-scale datasets, the de-facto choice is vanilla full fine-tuning, which tunes all the model parameters with the downstream data. To reduce memory footprint and avoid overfitting, an emerging trend is to tune a small proportion of the model parameters while freezing the majority ones with Parameter-Efficient Fine-Tuning (PEFT) [27, 28, 31]. However, both full fine-tuning and PEFT target an exclusive specific resource budget for each downstream task, while in reality, we often need to deploy multiple models simultaneously to meet various resource demands. This makes us ponder: *What is an effective and efficient way to obtain a palette of fine-tuned models meeting different resource constraints?*

A natural approach is to compress a well-trained large model into numerous smaller ones [11, 21, 29, 36]. However, the computational cost grows linearly with the number of deployment scenarios. Following the once-for-all network [8, 20], a few works first pre-train a weight-sharing over-parameterized supernet, then adapt the supernet to the downstream tasks. Although promising, training supernets at scale with large datasets requires prohibitive computational resources, *e.g.*, thousands of GPU training hours, which is practically infeasible.

On the other hand, there are many open-source large models [5, 23, 64] from communities such as HuggingFace [72] that are pre-trained on large-scale datasets and ready to be downloaded. Inspired by the success of model stitching [4, 12, 38], recently, Stitchable Neural Networks (SN-Net) [51] has been proposed to stitch pre-trained models (anchors) of the same family to quickly obtain a set of candidate new networks (stitches) with different accuracy-efficiency trade-offs of a wide FLOPs range. However, the scope of SN-Net is limited to the pre-training classification task on the same source domain [56]. When directly adapting the standard approach of SN-Net to a target domain, it faces new challenges. Specifically, the straightforward way to employ SN-Net in task adaptation typically requires three stages: first adapting anchors individually to the target domain, followed by stitching the adapted anchors, and finally evaluating all stitches to find and deploy the ones on the Pareto Frontier. Such a three-stage adaptation pro-

†Corresponding author. E-mail: bohan.zhuang@gmail.com
[1]Source code will be released at https://github.com/ziplab/Stitched_LLaMA.

Table 1. Comparison of frameworks for obtaining a palette of networks to meet different hardware efficiency constraints. Supernet-based methods NAT [47] and TOFA [35] necessitate a supernet training stage plus long deployment GPU hours and are restricted to small FLOPs ranges. SN-Net [51] employs model stitching to achieve a wide FLOPs range with significantly reduced deployment GPU hours. Our ESTA framework further reduces the deployment GPU hours, fine-tuning memory, and trainable parameters, and is scalable to produce LLM stitches with billions of parameters (Section 5.2). The deployment GPU hours and fine-tuning memory for both SN-Net and ESTA are measured on a single NVIDIA GeForce RTX 3090 GPU when stitching ViT-Ti/S/B anchors [16] with batch size 64. * indicates data is inferred from the training recipe of TOFA's supernet architecture FBNetV3 [14].

Method	Adapting supernet	Deployment GPU hours	Fine-tuning memory	Trainable parameter #	FLOPs range (G)	LLM friendly
NAT [47]	✗	>1,000	-	>10M	$[0.2, 0.6]$	✗
TOFA [35]	✓	>1,000*	-	>10M*	$[0.2, 2.5]$	✗
SN-Net [51]	✗	19.3	13,235M	124.2M	$[1.3, 17.6]$	✗
ESTA (ours)	✗	5.0	9,685M	4.6M	$[1.3, 17.6]$	✓

cess is expensive and sub-optimal. Moreover, adapting SN-Net typically needs to load and optimize all parameters of multiple models, leading to a daunting training-time memory cost. For instance, stitching the two smallest models in the LLaMA family [64, 64] with SN-Net under standard training settings (32-bit floating point parameters and Adam optimizer on a single GPU) requires more than 80G Video RAM that cannot fit into high-end GPUs such as NVIDIA A100. SN-Net also requires saving a separate instance of the model family for each task, making the storage grow linearly with the number of deployed applications.

Therefore, in this paper, we introduce a novel Efficient Stitchable Task Adaptation (ESTA) framework. Specifically, built upon SN-Net [51], ESTA introduces two main designs to overcome the aforementioned issues. *First*, to reduce the memory and storage costs, we tailor a Parameter-efficient Stitch fine-Tuning (PST) method, which incorporates the representative sparse fine-tuning technique LoRA [28] to keep the anchors and stitching layer weights frozen, while approximating their updates with trainable low-rank decomposition matrices. In addition to stitch-agnostic LoRA modules, we further introduce stitch-specific bias terms to alleviate the conflict among different stitches. *Second*, we propose a simple one-stage deployment pipeline to simultaneously adapt and stitch the pre-trained anchors to the target domain. Moreover, we propose to estimate and accumulate an importance score for each stitch via the saliency pruning criterion [37]. With the importance scores, we develop a novel stitch-sampling method to assign important stitches with higher sampling probabilities. More importantly, after fine-tuning, the stitch importance scores can be directly used to infer the best ones at the Pareto frontier for deployment. Table 1 compares different efficient frameworks for diverse deployments.

Overall, our paper has the following major contributions. 1) This is a pioneering work to investigate the problem of efficient fine-tuning of SN-Net for task adaptation. Our solution ESTA is a successful endeavor that delivers dozens of ready-to-deploy vision models up to 17.6G FLOPs in 5 GPU hours as well as multiple chatbot models with billions of parameters by stitching LLaMA models [64, 65] for the instruction-following task. 2) We devise a parameter-efficient stitch fine-tuning method that incorporates trainable stitch-agnostic LoRA modules and stitch-specific bias terms, which largely reduces the fine-tuning memory footprints while alleviating the interference issue among stitches. Moreover, we streamline a simple one-stage deployment pipeline with a novel task-specific stitch sampling strategy that greatly reduces the deployment time and improves Pareto frontiers. 3) Extensive experiments conducted on 25 downstream visual recognition tasks demonstrate that when stitching ViT-Ti/S/B [16], our ESTA achieves remarkable performance improvements compared to the direct SN-Net adaptation counterpart while utilizing significantly fewer trainable parameters.

2. Related Work

Model stitching. Model stitching [4, 38] targets connecting the bottom layers of one network to the top layers of another with a stitching layer. Following [38], model stitching is first employed to measure the similarities for the inner representations learned by deep neural networks [4, 12, 25]. Previous works [4, 12] observe that trained networks with different initializations can be effectively stitched without incurring significant performance drop. Based on this observation, DeRy [76] stitches the blocks dissected from different pre-trained models and assembles a new model based on their quantified similarity for better performance. Herdt *et al*. [24] propose interpreting the inner representations of a deep network by stitching it with a pre-trained GAN generator. Recently, from the perspective of flexible deployment, SN-Net [51] stitches models of different sizes within a model family to cheaply produce numerous new networks with diverse accuracy and efficiency trade-offs in the pre-training task. In contrast to [51], our work specifically targets obtaining a palette of fine-tuned models for the task adaptation setting. To do so, we make key redesigns including a parameter-efficient stitch fine-tuning method and a simple one-stage deployment pipeline to overcome the efficiency challenges in SN-Net adaptation.

Parameter-efficient fine-tuning. Parameter-efficient fine-tuning [27, 28, 31, 40] is a powerful alternative to vanilla full tine-tuning, which updates only a small number of parameters while freezing the majority ones. Freezing the majority of parameters allows us to optimize storage and reduces the burden on training GPU memory, as there is no need to store their gradients or other training time statistics. Recent research freezes the vast majority of parameters while fine-tuning either the parameters that are inherited in the backbone [77, 80] or additionally added, in the form of learnable prompts [31, 39], low-rank bottleneck layers [27], learnable scaling and shifting factors [41], and separated small networks [61]. Meanwhile, PEFT has been employed in numerous downstream applications. For instance, while freezing the most parameters, Polyhistor [45] learns a hyper-network to generate adapter weights [27] for multi-task adaptation, LLaMA-Adapter [83] inserts learnable prompts into Transformer layers for the instruction-following task, and ControlNet [82] employs the zero-initialized convolutions to fine-tune diffusion generative models [55]. However, these works target only one exclusive resource budget and are not scalable to diverse deployment scenarios. Unlike other methods, our ESTA is crafted to yield multiple fine-tuned models with diverse capacities while jointly incorporating stitch-specific and stitch-agnostic lightweight trainable parameters.

Fine-tuning with multiple models. The presence of more large-scale models has unleashed the potential to utilize multiple models instead of a single one during fine-tuning. One line of work selects the best pre-trained model in the model zoo to fine-tune. To identify the best model, prior studies quantize the transferability of the pre-trained models ahead of fine-tuning by estimating their accuracy on the downstream tasks [3, 17, 33] or the generalization capability to mitigate the domain gaps [6, 49, 66, 78]. Another way is to do model selection using an efficient online learning regime [74]. Considering the diversity among the pre-trained models, research efforts have been devoted to effectively exploit their knowledge with feature aggregation [42, 57, 58], model merging [13, 30, 48, 73], or a mixture-of-experts architecture [59]. In contrast to previous works that merely target better performance, our work employs off-the-shelf pre-trained model families to produce plentiful new models for diverse deployment requirements.

3. Preliminaries

3.1. Model Stitching

Considering an L-layer feed-forward artificial neural network $f_{\theta_1} : \mathcal{X} \to \mathcal{Y}$ parameterized by θ_1, that maps any input from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, f_{θ_1} can be denoted as a composition of functions that $f_{\theta_1} = f_L \circ \cdots \circ f_1$, where $\circ$ denotes the function composition. In model stitching, f_{θ_1} can be split up at the l-th layer into two portions of functions where $l \in [1, L-1]$. Given any input $\boldsymbol{X} \subseteq \mathcal{X}$, the first portion of the head functions maps $\boldsymbol{X}$ to activations $\boldsymbol{X}_l$ at layer l, *i.e.*, $H_{\theta_1,l}(\boldsymbol{X}) = f_l \circ \cdots \circ f_1 = \boldsymbol{X}_l$. The second portion of the tail functions maps $\boldsymbol{X}_l$ to the final output, *i.e.*, $T_{\theta_1,l}(\boldsymbol{X}_l) = f_L \circ \cdots \circ f_{l+1}$.

Given another pre-trained M-layer artificial neural network f_{θ_2} parameterized by θ_2 that is split up at the m-th layer that $m \in [2, M]$, we can then employ a stitching layer $S : \mathcal{A}_{\theta_1,l} \to \mathcal{A}_{\theta_2,m}$ to map between the two activation spaces at layers l and m. In this case, we obtain a new network parameterized by ϕ and the stitching layer parameters, *i.e.*,

$$F_{\phi,S}(\boldsymbol{X}) = T_{\theta_2,m} \circ S \circ H_{\theta_1,l}(\boldsymbol{X}), \tag{1}$$

where ϕ consists of partial parameters from both θ_1 and θ_2.

3.2. Stitchable Neural Network

In Stitchable Neural Network (SN-Net) [51], let a pre-trained model family of size Z be $\mathcal{Z} = \{f_{\theta_z}\}_{z=1}^{Z}$, where θ_z is the model parameter of the z-th model. The goal is to derive additional N candidate new networks $\mathcal{N} = \{F_{\phi_n,S_n}\}_{n=1}^{N}$ to adapt to various resource constraints. To do so, SN-Net first selects a pair of pre-trained models (anchors) from $\mathcal{Z}$ and then stitches them with Eq. (1) at different layer indexes to get N stitches. Then, SN-Net samples the stitches randomly and jointly optimizes them. An overview of SN-Net is depicted in Figure 1 (a). Since the anchors vary in scale, the newly assembled stitches have diverse performances and complexities. Importantly, SN-Net gives practical principles to design the space of $\mathcal{N}$ that one should 1) stitch a pair of nearest anchors in terms of model complexity in a model family; 2) stitch the head of a faster and smaller anchor to the tail of a larger and slower anchor. Unless specified otherwise, we adopt these as the default experimental settings in this paper.

4. Method

In this section, we introduce our ESTA framework, which consists of two major components: *parameter-efficient stitch fine-tuning* (PST) and *a simple one-stage deployment pipeline*. PST is tailored to cheaply fine-tune a palette of stitches (Section 4.1 and Figure 1 (b)). The one-stage deployment pipeline aims to simultaneously save deployment time and improve adaptation performance (Section 4.2).

4.1. Parameter-efficient Stitch Fine-tuning

To address the problems of large storage and memory consumption when adapting SN-Net to downstream tasks with full fine-tuning, we introduce a parameter-efficient stitch fine-tuning (PST) method as depicted in Figure 1 (b). Our basic idea is to adapt the representative sparse fine-tuning method LoRA [28] for SN-Net fine-tuning.

LoRA on Transformer layers. Specifically, let any pre-trained weight matrix in the multi-head self-attention layer

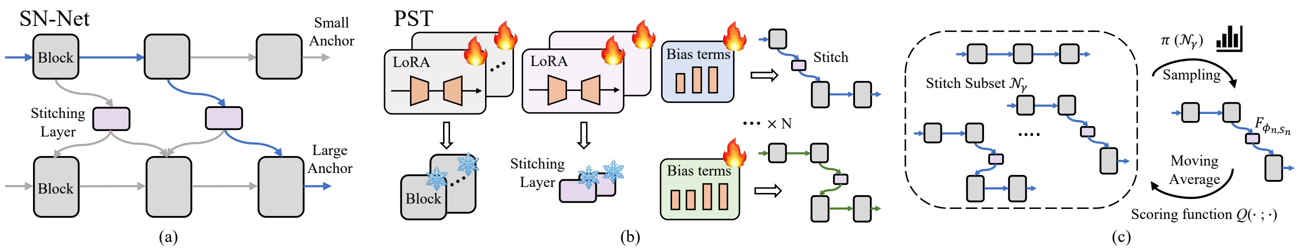

Figure 1. (a) Illustration of Stitchable Neural Network [51]. With two anchors from the same model family, SN-Net connects the early layers of the smaller one to the latter layers of the larger one with stitching layers to obtain a set of new networks with different performance-efficiency trade-offs, *e.g*., the path in Blue. (b) Overview of our PST method tailored for fine-tuning a palette of stitches, which integrates stitch-agnostic LoRA modules with stitch-specific bias terms, aiming to promote diverse representations among stitches while maintaining low trainable parameters. (c) Overview of our task-specific stitch sampling. We estimate the importance scores of the stitches with a scoring function $Q(\cdot,\cdot)$ and accumulate them as global statistics with moving averages. For a resource constraint γ, we sample with a categorical distribution $\pi(\mathcal{N}_\gamma)$ that is parameterized by the normalized importance scores so as to assign the important stitches with higher sampling probabilities. After fine-tuning, we directly deploy the stitches with the highest scores to avoid the costly evaluation stage.

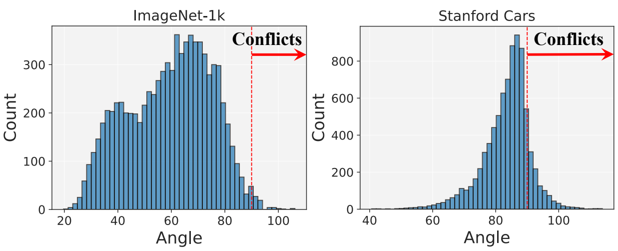

Figure 2. Distribution of pair-wise gradient angles among stitches when updating shared weights at fine-tuning iteration 600. We highlight angle 90° with a dashed red line. For simplicity, we show the gradient angles among the combined query, key, and value projection matrices for a total of 32 stitches when stitching ViT-Ti and ViT-S anchors. Generally, the gradient angles are larger in the target domain Stanford Cars [18] than in the source domain ImageNet-1k [56].

be $\boldsymbol{W} \in \mathbb{R}^{d\times k}$, we freeze $\boldsymbol{W}$ and insert trainable low-rank decomposition matrices $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{r\times k}$, where r is the pre-defined rank that $r \ll \min(d,k)$. In this way, the updated version of $\boldsymbol{W}$ can be formulated as

$$\boldsymbol{W}' \leftarrow \boldsymbol{W} + \boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}. \tag{2}$$

We follow [28] to respectively use Gaussian and zero initializations for $\boldsymbol{W}_{\text{up}}$ and $\boldsymbol{W}_{\text{down}}$, so that $\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$ is zero at the beginning of fine-tuning.

LoRA on stitching layers. We further extend the low-rank weight update to stitching layers. In SN-Net, a stitching layer is designed as a full-rank transformation matrix. Without loss of generality, we show the low-rank update of stitching layer S_n that stitches f_{θ_1} and f_{θ_2} at layers l and m, respectively. We first follow the initialization of stitching layers in [12, 51] to align the activations at layers l and $m-1$. Let the activations be $H_{\theta_1,l}(\boldsymbol{X}) \in \mathbb{R}^{b\times d_1}$ and $H_{\theta_2,m-1}(\boldsymbol{X}) \in \mathbb{R}^{b\times d_2}$ with sequence length b and feature dimensions d_1 and d_2. The stitching layer is parameterized by a transformation matrix $\boldsymbol{M} \in \mathbb{R}^{d_1\times d_2}$, which is initialized by solving a least squares problem, *i.e*., $\boldsymbol{M} = H_{\theta_1,l}(\boldsymbol{X})^\dagger H_{\theta_2,m-1}(\boldsymbol{X})$, where $H_{\theta_1,l}(\boldsymbol{X})^\dagger$ is the Moore-Penrose pseudoinverse of $H_{\theta_1,l}(\boldsymbol{X})$. We find that such initialization already has impressive representational capacity on the target domain and they can be updated with low-rank decomposition without hurting the performance too much. Accordingly, we update $\boldsymbol{M}$ similar to updating $\boldsymbol{W}$ in Eq. (2) by approximating its update with two learnable low-rank matrices with the same initializations.

Although the low-rank updates significantly enhance parameter efficiency, the low-rank essence [2] largely limits the network capacity. Particularly, in task adaptation, we often observe conflicting updates on the shared weights among the stitches. Figure 2 gives an example, where we stitch pre-trained anchors on ImageNet-1k [56] and Stanford Cars [18] and visualize the distribution of pair-wise gradient angles among different stitches when updating shared weights. We find that stitches agree less and conflict more (*i.e*., with more large angles) on the target domain Stanford Cars compared to the source domain ImageNet-1k.

Stitch-specific bias. To address the above issue, we further introduce stitch-specific bias terms as depicted in Figure 1 (b). Particularly, for a linear layer parameterized by $\boldsymbol{W}$ and a bias term $\boldsymbol{b}$, we add bias term $\boldsymbol{b}^s$ when optimizing the s-th stitch, for which the output activation becomes $\boldsymbol{W}\boldsymbol{X} + \boldsymbol{b} + \boldsymbol{b}^s$. We employ a set of distinct bias terms at different layers for each stitch. In this way, the stitches are encouraged to learn distinct feature representations for different resource requirements. To restrict the number of trainable parameters, following [28], we only apply PST to self-attention layers while freezing feed-forward layers.

4.2. One-stage Deployment Pipeline

Simultaneously adapt and stitch anchors. As aforementioned, given a model family $\mathcal{Z}$ pre-trained in the source domain, directly adapting SN-Net to a target domain typically involves three stages. First, adapting each anchor to the target domain $\mathcal{D}$ by solving $\theta_z{}^* = \operatorname{argmin}_{\theta_z}\mathcal{L}(\theta_z;\mathcal{D})$. In the second stage, at each training iteration, SN-Net samples a stitch and optimizes it with objective $\mathcal{L}(\phi_n^*, S_n; \mathcal{D})$, where ϕ_n^* is the set of SN-Net parameters for stitch F_{ϕ_n, S_n} which has been optimized once in the first stage. In the final stage, all optimized stitches need to be evaluated to identify the ones on the Pareto frontier for diverse resource constraints. Such a three-stage approach is expensive and sub-optimal (see more discussions in the supplementary material). Thus, we propose to adapt and stitch the anchors simultaneously within one stage, by directly optimizing $\mathcal{L}(\phi_n, S_n; \mathcal{D})$ for each sampled stitch. Moreover, we introduce a novel task-specific stitch-sampling method to allow the promising stitches more likely to be sampled, based on a stitch importance score. After fine-tuning, the stitches with the highest scores are naturally selected for deployment without the need for the above final stage evaluation.

Task-specific stitch sampling. SN-Net assumes that all stitches are equally important and performs random (uniform) sampling during training. However, as recognized by prior research [43, 70] that random sampling causes a gap between training and deployment as only the best stitches on the Pareto frontier are actually deployed. Moreover, since the importance of the pre-trained weights varies in different downstream tasks [19, 22, 75], we argue that the performance of stitches that are parameterized by these weights is also different across these tasks.

To this end, we propose to assign the important stitches that are likely to be on the Pareto frontier with higher sampling probabilities during fine-tuning to ensure that they are optimized sufficiently. Specifically, we first estimate the importance of each stitch with a data-dependent saliency pruning metric SNIP [37], which measures the importance with the first-order gradient information with barely any extra computational cost. When the n-th stitch is sampled, we can get its importance score $Q(F_{\phi,S_n}, \boldsymbol{X})$ with the scoring function $Q(\cdot,\cdot)$ [37]. To obtain robust importance scores for sampling, we accumulate scores with moving average during training, *i.e.*,

$$q_n^t \leftarrow \eta q_n^{t-1} + (1-\eta) Q(F_{\phi_n,S_n}, \boldsymbol{X}), \tag{3}$$

where q_n^t and q_n^{t-1} are the importance score at the t-th and $(t-1)$-th iteration, respectively, and $\eta \in [0, 1)$ is the momentum coefficient. In this way, we can get stable importance scores. After a warm-up period that employs uniform sampling to accumulate the scores, we assign the important stitches with higher sampling probabilities for the rest of the fine-tuning epochs. To do so, we uniformly divide the resource constraint range of our stitches into several intervals and sample an interval γ with uniform sampling. Accordingly, we can obtain a subset of stitches $\mathcal{N}_\gamma$ whose resource constraints belong to γ and their corresponding importance scores $\mathcal{Q}_\gamma$. We can then define a categorical distribution based on the normalized importance scores, *i.e.*, $\pi(\mathcal{N}_\gamma) = \mathrm{softmax}(\mathcal{Q}_\gamma)$ and finally get the sampled stitch $F_{\phi_n,S_n} \sim \pi(\mathcal{N}_\gamma)$. We show that our task-specific stitch sampling improves the performance in Section 5.3.

Additionally, with the sampled stitch, SN-Net employs knowledge distillation with a RegNetY-160 [54] pre-trained teacher to improve its performance. However, pre-trained teachers on downstream tasks are often unavailable. Therefore, in each training iteration, we always sample and train the largest anchor (teacher) and transfer its knowledge to the sampled stitch (student) similar to inplace distillation [79]. In practice, we employ the hard-label distillation [26, 63].

Reuse stitch importance scores for deployment. After fine-tuning, with the set of accumulated importance scores $\mathcal{Q}_\gamma$ for all stitches, we directly use them to select the stitches with the highest scores to deploy, which eliminates the need for the costly evaluation of the final stage of the straightforward SN-Net adaptation solution. We empirically observe that the deployed stitches are mostly on the Pareto frontier in Section 5.3, suggesting our estimation of the important stitches is accurate. Furthermore, in the supplementary material, we show that the important stitches differ across different tasks, proving the need for task-specific deployment strategies, as highlighted in [19, 22, 75]. It's worth noting that although the SNIP metric has been widely employed as a one-shot metric to efficiently estimate network performance [1, 69], we extend it to estimate stitch importance for better training and selection.

5. Experiments

5.1. Visual Recognition

We evaluate the effectiveness of our method on a total of 25 downstream visual recognition tasks, including fine-grained visual classification (FGVC) benchmark, common visual classification benchmark CIFAR-100 [34], and VTAB-1k [81] benchmark. FGVC benchmark contains NABirds [67], CUB-200-2011 [68], Stanford Cars [18], Stanford Dogs [32], and Oxford Flowers [50] tasks and the VTAB-1k benchmark includes 19 tasks in different domains, each of which has 800 training samples. For visual recognition experiments, we stitch ImageNet-21k pretrained ViT-Ti/S/B models [16] from [60].

Implementation details. We employ the stitching space and settings (kernel size as 2 and stride as 1) in [51] for downstream visual recognition tasks. We uniformly divide the resource constraints supported by the stitches into around 15 intervals and deploy one stitch within each interval. We set the hyper-parameter r for the rank of weight updating in the self-attention layers to be 32, 16, and 8 for ViT-Ti, ViT-S, and ViT-B, respectively. We also set $r = d_1//4$ universally for all stitching layers and η in Eq. (3) as 0.9 by grid search. We follow [31] to fine-tune 100 epochs on each task and select the other hyper-parameters and augmentation methods. We include more implementation details in

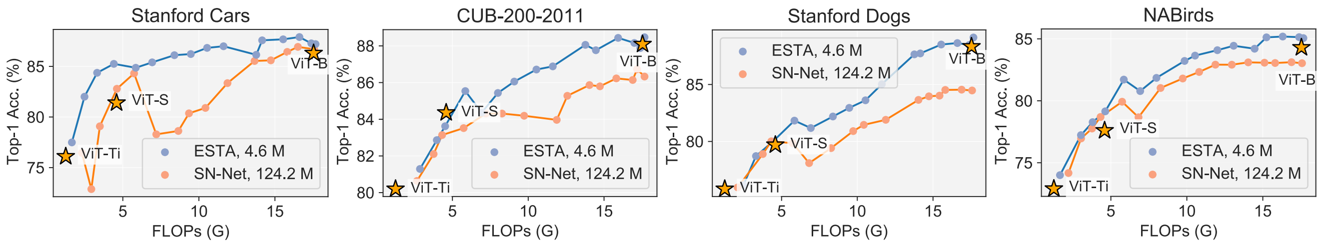

Figure 3. Performance comparisons with SN-Net [51] for adapting ViT-Ti/S/B pre-trained on ImageNet-22k [15] to Stanford Cars [18], CUB-200-2011 [68], Stanford Dogs [32], and NABirds [67]. We denote individually fine-tuned anchors as yellow stars. We also show the number of trainable parameters.

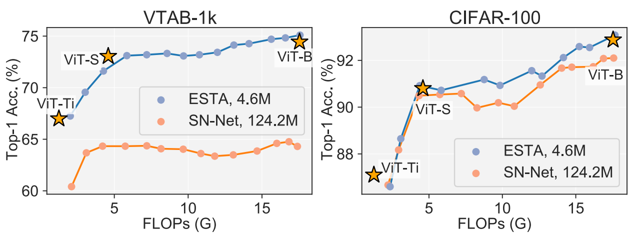

Figure 4. Performance comparisons with SN-Net [51] for adapting ViT-Ti/S/B pre-trained on ImageNet-22k [15] to VTAB-1k [81] and CIFAR-100 [34]. We denote individually fine-tuned anchors as yellow stars and also show the number of trainable parameters.

Table 2. Relative response quality to Alpaca-LoRA 7B [62] on Vicuna Bench [10]. Our Stitched LLaMA successfully interpolates the answer quality between Alpaca-LoRA 7B and 13B.

	Alpaca-LoRA		Stitched LLaMA				
Parameter # (B)	6.7	13.0	6.9	7.8	11.5	12.5	13.0
Quality (%)	100	123	113	114	115	120	123

the supplementary material.

Main results. We compare the performance of our ESTA with SN-Net and the individual anchors fine-tuned with LoRA [28] on various datasets. The FLOPs-accuracy curves are visualized in Figures 3 and 4. We report the averaged results over the three task groups on VTAB-1k following [31]. Overall, the stitches obtained by our ESTA exhibit smooth FLOPs-accuracy trade-off curves consistently across all datasets. Notably, the curve for our ESTA outperforms SN-Net by significant margins with much fewer trainable parameters. We show that the inferior results of SN-Net on downstream tasks are led by the sub-optimal designs of fine-tuning all model parameters and a three-stage adaptation pipeline in Section 5.3.

We also observe that the stitches even outperform the anchors with comparable or lower FLOPs in many cases. For instance, stitches outperform individually fine-tuned ViT-S/B anchors by clear margins on NABirds, CUB-200-2011, Stanford Cars, and Stanford Dogs datasets with comparable computational complexity. This phenomenon is consistent with the observations made in [52, 79]. We conjecture that weight sharing among the stitches serves as a strong regularization to improve their generalization. We show more results on few-shot learning and stitching Convnets [46] in the supplementary material.

5.2. Instruction-following Task

We also evaluate the instruction-following capability when stitching the LLaMA family [64] with our ESTA framework. We adapt and fine-tune LLaMA-7B/13B on the 52K instruction-following dataset from [62] to obtain chatbot stitches of varying sizes. The 52K instruction-following data is generated from 175 human-written instruction-output pairs with [71]. Each sample in the dataset is formulated by a task description, the context of the task, and the answer that is generated by GPT-3.5 (text-davinci-003) [7].

Implementation details. We employ the stitching space and settings (kernel size as 1 and stride as 1) in [51] for the instruction-following task and introduce 40 stitches. We uniformly divide the resource constraints supported by the stitches into 6 intervals and deploy one stitch within each interval. We set η in Eq. (3) as 0.9. The rank r is set to 16 and $d_1//4$ for LoRA modules in self-attention and stitching layers, respectively. We employ Adam optimizer and fine-tune with batch size 128 for 10 epochs with gradient accumulation. During the generation stage, we employ top-p sampling to sample from the top subword candidates with the probability $p = 0.75$, conduct the beam search, and set the beam size as 4.

Results. We term the stitches obtained by our ESTA framework as Stitched LLaMA and compare them with Alpaca-LoRA 7B and 13B that respectively freeze the LLaMA 7B and 13B backbones and update the self-attention weights with LoRA modules [28]. For quantitative results reported in Table 2, we observe that similar to the visual recognition tasks, our Stitched LLaMA with different numbers of parameters generates answers with plausible quality in-between Alpaca-LoRA 7B and 13B. Figure 5 gives qualitative comparisons. Our Stitched LLaMAs produce reasonable responses, suggesting the potential of our ESTA for adapting large language models (LLMs) for more downstream language tasks. More examples can be found in the supplementary material.

5.3. Ablation Study

Effect of parameter-efficient stitch fine-tuning. We investigate the effectiveness of our PST on five FGVC tasks and visualize the averaged results in Figure 6 (a) and (b).

Instruction: Tell me about alpacas.

Alpaca-LoRA 7B (0.035 s/token): Alpacas are members of the camelid family, which also includes llamas and camels. They are native to the Andes Mountains in South America, where they have been domesticated for thousands of years. Alpacas are known for their soft, luxurious fiber, which is used to make clothing, blankets, and other textiles. They are also raised for their meat, which is considered a delicacy in some parts of the world. Alpacas are gentle, intelligent animals, and they are prized for their docile nature and ability to thrive in a wide range of climates.

Stitched LLaMA 8B (0.037 s/token): Alpacas are herbivorous mammals that belong to the family Camelidae, which also includes camels and camelids. They are native to the South American continent and are commonly found in Peru, Argentina, Bolivia, and Chile. Alpacas are smaller than their cousin, the camel, and are typically about 80-90 inches tall when fully grown. They are known for their soft, shimmering, and light-weight fibers, which are used to make clothing, carpets, and other textile products. Alpacas are herbivores, and their diet consists mainly of grasses, leaves, and straw. They are social animals and live in herds of up to 100 alpacas, led by a dominant male. Alpacas are known for their calm and gentle nature, and they are often used as livestock for their soft, shimmering fibers.

Stitched LLaMA 11B (0.041 s/token): Alpacas are native to South America and are members of the camelid family. They are known for their soft, fluffy coats and their gentle nature. Alpacas are herbivores, meaning they feed on grass and other vegetation. They have a gestation period of approximately 11 months and usually give birth to a single baby, although twins and triplets are not uncommon. Alpacas are social animals and live in herds, led by a dominant male. They communicate with each other using a variety of vocalizations, including whistles, grunts, and barks. Alpacas are intelligent animals and are known for their curiosity and playful nature. Alpacas have been domesticated for thousands of years and are used for a variety of purposes, including as pets, for their fiber, and for their meat. Alpaca fiber is prized for its softness and warmth, making it a popular material for clothing and blankets. Alpaca meat is also considered a delicacy in some parts of the world. Overall, alpacas are gentle, intelligent animals that have been domesticated for thousands of years. They are valued for their fiber, meat, and companionship.

Instruction: Can you help me write a resignation letter to my current employer, while leaving on good terms and expressing gratitude for the opportunities provided?

Alpaca-LoRA 7B (0.035 s/token): Dear [Name of Employer], I am writing to inform you that I am resigning from my position as [Job Title] with [Company Name]. It has been a pleasure working with you and I am grateful for the opportunities you have provided me. I wish you and the company all the best in the future. Sincerely, [Your Name]

Stitched LLaMA 8B (0.037 s/token): Dear [Employer], I am writing to inform you that I am resigning from my position at [Company Name] with a heavy heart. I have enjoyed my time working with you and the team, and I am grateful for the opportunities that have been provided to me. I would like to express my sincere appreciation for the training, support, and opportunities that have been provided to me during my time at [Company Name]. I have learned a great deal and have grown personally and professionally. Please accept my best wishes for the continued success of [Company Name] and the team. I wish you all the best in the future. Sincerely, [Your Name]

Stitched LLaMA 11B (0.041 s/token): Dear [Employer], I am writing this letter to inform you of my decision to resign from my position at [Company]. It has been a great pleasure to work for [Company] and I have learned a lot during my time here. I would like to express my gratitude for the opportunities that have been provided to me and for the support and guidance that I have received from my colleagues and supervisors. I have grown both professionally and personally during my time at [Company] and I am grateful for the experiences I have had. I am leaving on good terms and would be happy to provide a positive reference for [Company] in the future. If there is anything I can do to assist with the transition, please do not hesitate to reach out. Thank you for the opportunity to work for [Company] and I wish you all the best in your future endeavors. Sincerely, [Your Name]

Figure 5. Instruction-following comparison between Stitched LLaMA obtained by our ESTA and the Alpaca-LoRA 7B fine-tuned with LoRA [28].

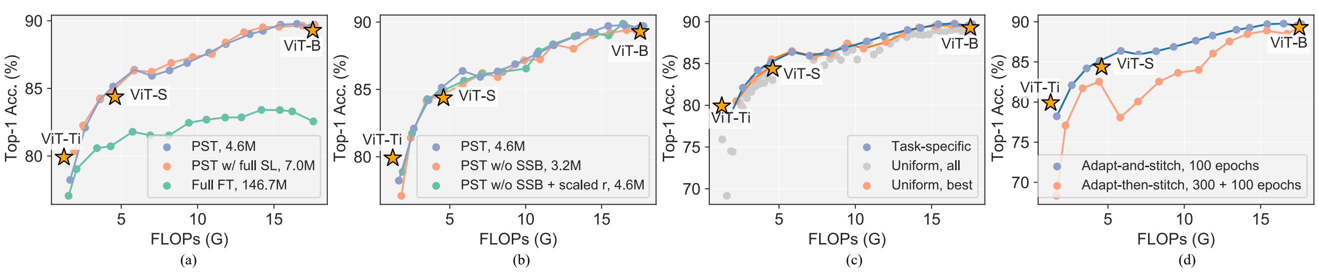

Figure 6. Ablation studies with results averaged over five FGVC tasks. (a) Effect of our stitch-agnostic LoRA modules. "Full FT" represents full fine-tuning of all the model parameters. "PST w/ full SL" employs fully fine-tuned stitching layers. We also show the number of trainable parameters. (b) Effect of our stitch-specific bias terms. "SSB" and "scaled r" represent stitch-specific bias terms and scaling the rank hyperparameter to reach 4.6M trainable parameters, respectively. (c) Effect of our task-specific stitch sampling. "Uniform, all" and "Uniform, best" represent all the stitches fine-tuned with uniform sampling and their best ones in each resource constraint interval on the Pareto frontier, respectively. (d) Effect of our strategy to simultaneously adapt and stitch anchors. Our strategy "Adapt-and-stitch" takes 100 epochs for fine-tuning. In contrast, "Adapt-then-stitch", as a straightforward approach to apply SN-Net, first individually adapts each anchor for a total of 300 epochs, and then fine-tunes SN-Net for another 100 epochs.

To recall, our PST keeps the majority of the model parameters frozen while introducing trainable stitch-agonistic low-rank decomposition matrices and stitch-specific bias terms. In Figure 6 (a), we observe that when employing LoRA to the multi-head self-attention layers, there are solid performance gains on all FLOPs ranges with significantly fewer trainable parameters, echoing the observations in [31, 41]. It is suggested that low-rank weight updates is a powerful alternative to full fine-tuning for stitching and adapting anchors and we speculate that low-rank weight update avoids overfitting under limited downstream data. We further employ low-rank weight updates to the stitching layers and save 2.4M more trainable parameters with a slight overall performance drop. We conjecture that the stitching layers for large pre-trained models are also in low intrinsic dimensions [2] and least-squares initialization [51] already provide a good initialization for them.

Figure 6 (b) shows that employing stitch-specific bias terms achieves better performance with affordable 1.4M extra trainable parameters. We also include the baseline that simply scales the low-rank dimension r to reach the same number of trainable parameters as our PST. This baseline is inferior to our PST. We speculate that the stitch-specific bias terms enable flexible adjustments of feature representations for different stitches, leading to enhanced performance.

Effect of task-specific stitch sampling. We investigate the effectiveness of our sampling strategy (introduced in Section 4.2) on the five FGVC tasks. The averaged results are visualized in Figure 6 (c). We compare our task-specific stitch sampling with the uniform one that is adopted in [51]. We empirically find that for uniform sampling, there is a clear performance gap between the best stitches that will be deployed and the others that will be dropped after training, suggesting the need for concentrating on the important stitches during training. To this end, our task-specific sampling strategy performs better than uniform sampling, indicating that our sampling strategy selects accurate task-specific important stitches. Most importantly, our strategy circumvents the need for the costly evaluation stage in the direct solution of adapting SN-Net for downstream tasks.

Effect of simultaneously adapting and stitching anchors. We study the effect of simultaneously adapting and stitching anchors strategy (introduced in Section 4.2) on five FGVC tasks. The averaged results are visualized in Figure 6 (d). We compare our approach with the direct SN-Net adaptation that first adapts each anchor individually for a total of 300 epochs before stitching them under the exact setting as ESTA. It can be seen that the direct SN-Net adaptation yields poor results in task adaptation. Fine-tuned individual anchors, especially the larger ones, are likely to overfit the limited downstream data [44, 53]. Accordingly, we speculate that the adapted anchor weights serve as a sub-optimal initialization for the stitches in the context of model stitching. On the other hand, our approach achieves better performance and greatly saves the fine-tuning cost.

Effect of the PEFT technique choices. We compare the effect of different PEFT technique choices and visualize the averaged results on the five FGVC tasks in Figure 7 left. We experiment to employ other PEFT techniques Adapter [27] and Adaptformer [9]. We follow their default setting to respectively insert bottleneck structures sequentially and in parallel to the feed-forward layers and fix the number of trainable parameters the same as ours for a fair comparison. We observe that the overall performance for different PEFT techniques is similar, suggesting that our ESTA framework is compatible with more PEFT technique choices. Since low-rank weight updates of LoRA [28] can be merged into

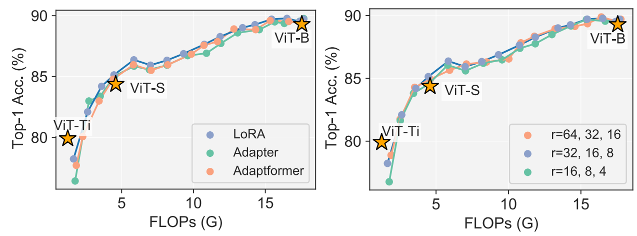

Figure 7. Left: effect of different PEFT technique choices. Right: effect of the rank hyper-parameter r in LoRA modules of self-attention layers, for which the three values represent the ranks in ViT-Ti, ViT-S, and ViT-B, respectively. The results are averaged on five FGVC tasks.

the backbone after training without extra inference computational cost, we employ LoRA by default.

Effect of the low-rank hyper-parameter r. We investigate the effect of different r which controls the rank when updating weights in self-attention layers on the five FGVC tasks. The averaged results on the five FGVC tasks are visualized in Figure 7 right. We find that setting r to be different values barely has any effect on the overall performance, suggesting our ESTA framework is robust to the rank hyper-parameter r. Since setting r to be 32, 16, and 8 for low-rank updating the self-attention weights in ViT-Ti, ViT-S, and ViT-B has slightly higher averaged performance among the stitches, we make it our default setting.

6. Conclusion and Future Work

In this paper, we have introduced a novel task adaptation framework to cheaply obtain a palette of fine-tuned networks via model stitching, supporting diverse efficiency-performance tradeoffs at runtime. Specifically, built on SN-Net, we have tailored a parameter-efficient stitch fine-tuning method, which learns lightweight stitch-agnostic LoRA modules and stitch-specific bias terms while keeping the majority of the parameters frozen. Our design significantly reduces fine-tuning memory and storage costs for downstream task adaptation. Moreover, we have devised a task-specific stitch sampling strategy to assign higher sampling probability to the important stitches during training, which simultaneously improves the Pareto frontiers and avoids a costly evaluation stage. Extensive experiments on 25 downstream visual recognition tasks and the instruction-following task have demonstrated the effectiveness of our proposed framework.

Limitations. Due to the constraints of computational resources, our experiments are limited to visual recognition and instruction-tuning tasks. In the future, we will explore adapting pre-trained model families to dense prediction and multi-modal tasks.

References

[1] Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas D. Lane. Zero-cost proxies for lightweight nas. In *ICLR*, 2021. 5

[2] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, 2021. 4, 8

[3] Haitham Bou Ammar, Eric Eaton, Matthew E Taylor, Decebal Constantin Mocanu, Kurt Driessens, Gerhard Weiss, and Karl Tuyls. An automated measure of mdp similarity for transfer in reinforcement learning. In *AAAIW*, 2014. 3

[4] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *NeurIPS*, 34:225–236, 2021. 1, 2

[5] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2022. 1

[6] Daniel Bolya, Rohit Mittapalli, and Judy Hoffman. Scalable diverse model selection for accessible transfer learning. *NeurIPS*, 34:19301–19312, 2021. 3

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020. 6

[8] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 1

[9] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS*, 2022. 8

[10] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. 6

[11] Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, pages 921–928, 2011. 1

[12] Adrián Csiszárik, Péter Kőrösi-Szabó, Akos Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations. *NeurIPS*, 34:5656–5668, 2021. 1, 2, 4

[13] Dong Dai and Tong Zhang. Greedy model averaging. *NeurIPS*, 24, 2011. 3

[14] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *CVPR*, pages 16276–16285, 2021. 2

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[16] 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. In *ICLR*, 2021. 2, 5

[17] Eric Eaton, Marie Desjardins, and Terran Lane. Modeling transfer relationships between learning tasks for improved inductive transfer. In *ECML PKDD*, pages 317–332. Springer, 2008. 3

[18] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *AAAI*, 2017. 4, 5, 6

[19] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *CVPR*, pages 4805–4814, 2019. 5

[20] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pages 544–560. Springer, 2020. 1

[21] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *NeurIPS*, 28, 2015. 1

[22] Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient tuning. In *ICCV*, 2023. 5

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 16000–16009, 2022. 1

[24] Rudolf Herdt, Maximilian Schmidt, Daniel Otero Baguer, Jean Le'Clerc Arrastia, and Peter Maass. Model stitching and visualization how gan generators can invert networks in real-time. *arXiv preprint arXiv:2302.02181*, 2023. 2

[25] Adriano Hernandez, Rumen Dangovski, Peter Y. Lu, and Marin Soljacic. Model stitching: Looking for functional similarity between representations. *NeurIPSW*, 2022. 2

[26] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NeurIPSW*, 2014. 5

[27] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, pages 2790–2799, 2019. 1, 3, 8

[28] 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, 4, 6, 7, 8

[29] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR*, 18(1):6869–6898, 2017. 1

[30] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018. 3

[31] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 1, 3, 5, 6, 7

[32] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *CVPRW*, 2011. 5, 6

[33] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, pages 2661–2671, 2019. 3
[34] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5, 6
[35] Achintya Kundu, Laura Wynter, Rhui Dih Lee, and Luis Angel Bathen. Tofa: Transfer-once-for-all. *arXiv preprint arXiv:2303.15485*, 2023. 2
[36] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *NeurIPS*, 2, 1989. 1
[37] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 2, 5
[38] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, pages 991–999, 2015. 1, 2
[39] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021. 3
[40] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021. 3
[41] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *NeurIPS*, 2022. 3, 7
[42] Iou-Jen Liu, Jian Peng, and Alexander G Schwing. Knowledge flow: Improve upon your teachers. In *ICLR*, 2019. 3
[43] Jing Liu, Jianfei Cai, and Bohan Zhuang. Focusformer: Focusing on what we need via architecture sampler. *arXiv preprint arXiv:2208.10861*, 2022. 5
[44] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 2023. 8
[45] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *NeurIPS*, 2022. 3
[46] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, pages 11976–11986, 2022. 6
[47] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *TPAMI*, 43(9):2971–2989, 2021. 2
[48] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *NeurIPS*, 35:17703–17716, 2022. 3
[49] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *ICML*, pages 7294–7305. PMLR, 2020. 3
[50] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, pages 722–729. IEEE, 2008. 5
[51] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Stitchable neural networks. In *CVPR*, 2023. 1, 2, 3, 4, 5, 6, 8
[52] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, pages 4095–4104. PMLR, 2018. 6
[53] Jo Plested and Tom Gedeon. Deep transfer learning for image classification: a survey. *arXiv preprint arXiv:2205.09904*, 2022. 8
[54] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, pages 10428–10436, 2020. 5
[55] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 3
[56] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115:211–252, 2015. 1, 4
[57] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 3
[58] Yang Shu, Zhi Kou, Zhangjie Cao, Jianmin Wang, and Mingsheng Long. Zoo-tuning: Adaptive transfer from a zoo of models. In *ICML*, pages 9626–9637. PMLR, 2021. 3
[59] Yang Shu, Zhangjie Cao, Ziyang Zhang, Jianmin Wang, and Mingsheng Long. Hub-pathway: Transfer learning from a hub of pre-trained models. *NeurIPS*, 2022. 3
[60] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *TMLR*, 2022. 5
[61] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *NeurIPS*, 2022. 3
[62] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023. 6
[63] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR, 2021. 5
[64] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1, 2, 6
[65] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2
[66] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *ICCV*, pages 1395–1405, 2019. 3
[67] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale

dataset with citizen scientists: The fine print in fine-grained dataset collection. In *CVPR*, pages 595–604, 2015. 5, 6

[68] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Tech. Rep. CNS-TR-2011-001, California Institute of Technology*, 2011. 5, 6

[69] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020. 5

[70] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *CVPR*, pages 6418–6427, 2021. 5

[71] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022. 6

[72] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *EMNLP*, pages 38–45, 2020. 1

[73] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, pages 23965–23998. PMLR, 2022. 3

[74] Hang Xu, Ning Kang, Gengwei Zhang, Chuanlong Xie, Xiaodan Liang, and Zhenguo Li. Nasoa: Towards faster task-oriented online fine-tuning with a zoo of models. In *ICCV*, pages 5097–5106, 2021. 3

[75] Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In *EMNLP*, 2021. 5

[76] Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. Deep model reassembly. *NeurIPS*, 35: 25739–25753, 2022. 2

[77] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *NeurIPS*, 27, 2014. 3

[78] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, pages 12133–12143. PMLR, 2021. 3

[79] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *ICCV*, pages 1803–1811, 2019. 5, 6

[80] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, pages 1–9, 2022. 3

[81] 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. 5, 6

[82] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 3

[83] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023. 3