Efficient Adaptation of Pre-trained Models: A Survey of PEFT for Language, Vision, and Multimodal Learning

Cheng Zhihao[1], Shufen Zhihao[1]

[1]Department of Computer Science and Technology, Fudan University, China

Abstract

The rapid scaling of pre-trained foundation models in natural language processing (NLP), computer vision (CV), and multimodal learning has led to growing interest in methods that can adapt these large models efficiently without incurring the full computational or storage costs of traditional fine-tuning. Parameter-Efficient Fine-Tuning (PEFT) methods address this challenge by modifying or introducing a small subset of learnable parameters while keeping the majority of the model frozen. In this survey, we present a comprehensive and systematic overview of the landscape of PEFT approaches. We categorize the main families of PEFT methods—including prompt tuning, adapter tuning, low-rank adaptation (e.g., LoRA), BitFit, and sparse updating—providing unified mathematical formulations, detailed comparative analyses, and extensive discussion of their theoretical underpinnings and empirical properties. We also explore implementation considerations, evaluation benchmarks, and real-world applications across language, vision, and multimodal domains. Finally, we highlight open challenges, interpretability gaps, and future research directions in this rapidly evolving field. Our goal is to serve as a foundation for researchers and practitioners seeking to understand, apply, or advance the state of the art in parameter-efficient adaptation of large-scale models.

Keywords: Parameter-Efficient Fine-Tuning, PEFT, Foundation Models, Large Language Models, Large Vision Models, Prompt Tuning, Adapter Tuning, LoRA, Low-Rank Adaptation, BitFit, Sparse Fine-Tuning, Transfer Learning, Multimodal Models, Efficient Adaptation, Model Compression, Continual Learning.

1 Introduction

The unprecedented growth of large-scale pre-trained models, such as large language models (LLMs) and vision-language models (VLMs), has led to significant performance improvements across a wide spectrum of tasks in natural language processing (NLP) and computer vision (CV). However, the immense size of these models, often comprising billions of parameters, presents a substantial challenge in terms of computational resources, storage, and energy consumption, especially during fine-tuning for downstream applications. *Parameter-Efficient Fine-Tuning* (PEFT) has emerged as a compelling paradigm to address these challenges. Rather than updating all model parameters during task-specific adaptation, PEFT strategies modify only a small subset of parameters or introduce lightweight modules, thereby reducing memory footprint and training cost while maintaining—or even surpassing—the performance of full fine-tuning. This paradigm is particularly attractive in the era of foundation models, where reusing a fixed backbone for multiple tasks is crucial for scalability and efficiency. In this survey, we provide a comprehensive overview of PEFT techniques applied to large language and vision models. We categorize PEFT methods based on their architectural and algorithmic principles, including adapter-based approaches, low-rank adaptation (LoRA), prompt tuning, and hybrid strategies [1]. We also highlight recent advances that push the boundaries of parameter efficiency, examine their empirical performance across diverse benchmarks, and discuss their practical considerations in deployment scenarios. Our goal is to present an A-to-Z guide to PEFT: from foundational concepts and motivating factors to detailed comparisons and open research challenges. By consolidating the growing body of work in this rapidly evolving area, we aim to provide researchers and practitioners with a clear and structured understanding of parameter-efficient fine-tuning, and its pivotal role in democratizing access to large-scale AI models.

2 Background and Motivation

Large-scale models, denoted as $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ represents the model parameters, have demonstrated remarkable performance on a wide range of tasks. However, the cost of training and fine-tuning such models scales with d, which can be prohibitively large (e.g., $d \approx 10^9$ to 10^{12} for modern LLMs and VLMs) [2]. In a conventional fine-tuning setup, all parameters $\boldsymbol{\theta}$ are updated to minimize a task-specific loss function $\mathcal{L}(\boldsymbol{\theta})$ on a target dataset $\mathcal{D}_{\text{target}} = \{(x_i, y_i)\}_{i=1}^{N}$ [3]. Formally:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \mathcal{L}(f_{\boldsymbol{\theta}}(x_i), y_i). \tag{1}$$

This approach is resource-intensive and impractical for settings with limited computational budgets or when deploying models across a fleet of devices [4].

2.1 Parameter-Efficient Fine-Tuning (PEFT)

PEFT techniques aim to decouple task-specific adaptation from full model retraining [5]. Instead of optimizing all parameters, PEFT methods introduce a small set of additional parameters $\boldsymbol{\phi}$ such that:

$$f_{\boldsymbol{\theta},\boldsymbol{\phi}}(x) = f_{\boldsymbol{\theta}}(x; \boldsymbol{\phi}), \tag{2}$$

where $\boldsymbol{\theta}$ remains mostly frozen and $\boldsymbol{\phi}$ is optimized. The number of trainable parameters $|\boldsymbol{\phi}| \ll |\boldsymbol{\theta}|$, often less than 1% of the full model size. Common PEFT strategies include:

- **Adapters:** Introduce bottleneck modules within the model layers.
- **Low-Rank Adaptation (LoRA):** Parameterize updates using low-rank matrices [6].
- **Prompt Tuning:** Optimize a set of task-specific embeddings prepended to the input.
- **BitFit:** Update only the bias terms in each layer.

2.2 Motivating Factors

Several factors motivate the use of PEFT in large-scale models:

1. **Scalability:** Reduces the cost of fine-tuning hundreds of models for different tasks or clients.
2. **Modularity:** Enables plug-and-play behavior by isolating task-specific parameters [7].
3. **Storage Efficiency:** Only the small parameter sets $\boldsymbol{\phi}$ need to be stored and shared [8].
4. **Continual Learning:** Facilitates lifelong learning without catastrophic forgetting.

2.3 Illustration of PEFT Paradigm

Figure 1 illustrates the fundamental concept behind PEFT using a transformer-based architecture [9].

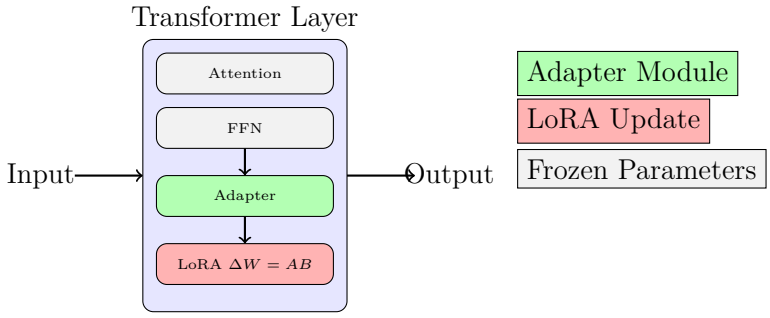

Figure 1: Illustration of PEFT using adapter and LoRA modules integrated into a transformer layer [10]. Only a small fraction of parameters are trained while the majority remain frozen.

3 Taxonomy of Parameter-Efficient Fine-Tuning Methods

To systematically understand the landscape of PEFT, we categorize methods based on how they modify or augment the pre-trained model. We broadly group PEFT techniques into four families:

1. **Adapter-based methods**
2. **Low-rank adaptation methods**
3. **Prompt-based methods**
4. **Weight selection and reparameterization**

Each family varies in design philosophy, trainable parameter footprint, and target application domains.

3.1 Adapter-Based Methods

Adapter methods insert small trainable modules Adapter_ϕ within or between the layers of a frozen backbone. Let a transformer block's hidden representation be $h \in \mathbb{R}^d$; an adapter augments it as:

$$\text{Adapter}_\phi(h) = W_{\text{up}} \cdot \sigma(W_{\text{down}} \cdot h), \tag{3}$$

where $W_{\text{down}} \in \mathbb{R}^{d_r \times d}$ and $W_{\text{up}} \in \mathbb{R}^{d \times d_r}$ are learnable projection matrices with $d_r \ll d$, and σ is a nonlinearity (e.g., ReLU or GELU). The residual addition to h ensures compatibility:

$$\tilde{h} = h + \text{Adapter}_\phi(h). \tag{4}$$

Popular examples include Houlsby Adapters and Compacter.

3.2 Low-Rank Adaptation Methods

Low-rank adaptation, popularized by LoRA, optimizes updates to existing weights via low-rank decompositions. For a weight matrix $W \in \mathbb{R}^{d \times d}$, LoRA freezes W and adds a low-rank delta:

$$W' = W + \Delta W, \quad \text{where } \Delta W = AB, \quad A \in \mathbb{R}^{d \times r},\ B \in \mathbb{R}^{r \times d}[11]. \tag{5}$$

The low-rank $r \ll d$ leads to a significant reduction in trainable parameters while enabling expressive task adaptation.

3.3 Prompt-Based Methods

Prompt tuning treats the model as a frozen feature extractor and prepends learnable tokens $\boldsymbol{p} = [p_1, \ldots, p_m]$ to the input. Let $x = [x_1, \ldots, x_n]$ be the original input; then the modified input becomes:

$$x' = [p_1, \ldots, p_m, x_1, \ldots, x_n], \tag{6}$$

which is passed through the model without altering its weights. Prefix tuning extends this idea by modifying key-value vectors inside attention blocks.

3.4 Weight Selection and Reparameterization

These approaches identify and tune a sparse subset of parameters, or reparameterize the model for efficient training [12]. BitFit, for instance, updates only bias terms:

$$\boldsymbol{\theta} = \{\text{bias terms}\}, \quad \text{all other parameters frozen.} \tag{7}$$

Other techniques like Diff Pruning and Lottery Tickets focus on sparsity and trainable subset selection.

4 Adapter-Based Fine-Tuning

Adapter-based fine-tuning methods constitute one of the most well-established and widely adopted families of parameter-efficient techniques [13]. The core idea is to freeze the vast majority of the pre-trained model parameters and insert small trainable modules—referred to as adapters—within the architecture. These adapters are typically placed between the layers of a transformer or convolutional neural network, depending on the modality[14]. Originally proposed in the context of transfer learning for multilingual models, adapters have proven to be remarkably flexible, enabling fine-tuning of large models on downstream tasks with minimal computational overhead [15]. A standard adapter module consists of a down-projection, nonlinearity, and up-projection structure, typically resembling a bottleneck MLP layer [16]. This configuration reduces the number of trainable parameters by a factor of r/d, where r is the bottleneck dimension and d is the hidden size of the model, while still providing enough expressivity to encode task-specific transformations [17]. A critical design decision lies in where these modules are inserted: either after the attention sub-layer, the feed-forward network, or both [18]. Empirical studies have found that placing adapters after each sub-layer (as proposed in the Houlsby Adapter architecture) yields more robust transfer across tasks. Let us denote the input to a transformer layer as $h \in \mathbb{R}^d$ [19]. In a traditional transformer architecture, h undergoes multi-head self-attention followed by a feed-forward network, both of which involve matrix multiplications with large weight matrices [20]. In the adapter-based paradigm, a task-specific adapter module Adapter_ϕ is inserted such that h is transformed into $\tilde{h} = h + \text{Adapter}_\phi(h)$ [21]. The adapter itself consists of two trainable matrices, $W_{\text{down}} \in \mathbb{R}^{d_r \times d}$ and $W_{\text{up}} \in \mathbb{R}^{d \times d_r}$, forming a bottleneck: $\text{Adapter}_\phi(h) = W_{\text{up}} \cdot \sigma(W_{\text{down}} \cdot h)$, where σ is a non-linear activation such as ReLU or GELU [22]. This design allows the adapter to act as a residual perturbation to the model's hidden representation, with the added benefit that the magnitude of this perturbation is tightly controlled by the small number of parameters in W_{down} and W_{up}. Crucially, since the rest of the model remains untouched, multiple adapters can be trained for different tasks and stored independently, enabling efficient multi-task adaptation and continual learning. This modular design allows a single pre-trained model to support an arbitrary number of downstream tasks by simply swapping adapter modules, which are typically in the order of a few megabytes compared to gigabyte-scale backbones. Beyond the basic bottleneck adapter, numerous variations have been proposed to improve expressivity, parameter sharing, or training stability [23]. For example, the Compacter architecture leverages hypercomplex multiplication to represent adapter weights more compactly, further reducing the parameter count without compromising performance [24]. Other extensions include the Parallel Adapter and the Parallel-Residual Adapter, which alter the topology of integration between the

adapter and the base model [25]. Instead of placing the adapter in series with the feed-forward or attention block, these variants allow adapters to run in parallel with the core layers, aggregating outputs before the residual connection [26]. This architectural choice can result in better gradient flow and improved stability during fine-tuning, especially in low-data regimes. Additionally, some works propose routing-based mechanisms where a shared pool of adapter modules is selectively activated based on the input or task identifier, enabling a mixture-of-experts flavor to adapter usage [27]. In multilingual or multitask scenarios, these dynamic adapters can route different language pairs or tasks through specialized pathways while maintaining a fixed backbone [28]. Adapter-based methods have also been extended to vision and multimodal models. In the vision domain, adapters have been integrated into vision transformers (ViTs) to enable scalable fine-tuning on diverse image classification and detection tasks. These visual adapters often use spatial projection and channel mixing to align with the spatial nature of image features [29]. In multimodal contexts, such as vision-language models (e.g., CLIP, Flamingo), adapters provide a lightweight way to specialize a shared model for downstream tasks like visual question answering or image captioning [30]. Here, task-specific adapters can be inserted into both the language and visual branches or even into the cross-modal fusion layers, enabling efficient task-specific adaptation while preserving the original model's broad capabilities [31]. This flexibility has proven especially useful in settings where a large foundation model is deployed across many applications with minimal retraining. The appeal of adapter-based methods also lies in their training dynamics. Since only a tiny portion of the model is updated, fine-tuning becomes substantially faster and more memory-efficient [32]. This property enables low-resource users to benefit from large-scale models using commodity hardware such as GPUs with 16–32 GB of VRAM [33]. Furthermore, adapters are naturally amenable to continual and federated learning, where clients can download a frozen base model and fine-tune a small adapter on their private data without communicating gradients for the full model [34]. This setup not only reduces communication cost but also enhances privacy, making adapter-based PEFT an attractive choice for real-world deployment scenarios. Moreover, the ability to switch adapter modules at inference time introduces a plug-and-play capability, where users can deploy different adapters for different tasks or domains without duplicating the underlying model weights. In summary, adapter-based fine-tuning methods offer a powerful and general-purpose framework for efficient model adaptation [35]. Their modularity, compactness, and strong empirical performance across modalities and tasks make them a foundational element of the PEFT landscape [36]. As the field continues to evolve, adapters are likely to remain a cornerstone of efficient transfer learning in both academic research and industrial applications. Future developments may include more dynamic and

context-aware adapters, tighter integration with retrieval-augmented generation, and further compression techniques to push the efficiency frontier even further.

5 Low-Rank Adaptation Methods

Low-rank adaptation methods, often grouped under the LoRA family, represent a powerful and elegant approach to parameter-efficient fine-tuning by exploiting the linear algebraic properties of neural network weight matrices. The foundational insight behind these methods is that the change in weights required to adapt a large pre-trained model to a new task often lies in a low-dimensional subspace [37]. Rather than updating the full weight matrix, which is both parameter- and computation-intensive, LoRA-style methods decompose the update into the product of two low-rank matrices. This strategy not only reduces the number of trainable parameters by orders of magnitude but also preserves the original pre-trained weights, ensuring that the base model's capabilities remain intact and reusable across tasks. Formally, consider a neural network layer characterized by a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, which transforms an input vector $x \in \mathbb{R}^{d_{\text{in}}}$ via Wx. In standard fine-tuning, W is updated to a new value $W' = W + \Delta W$, where ΔW is a dense matrix learned from scratch [38]. In LoRA, however, ΔW is parameterized as a low-rank matrix: $\Delta W = AB$, where $A \in \mathbb{R}^{d_{\text{out}} \times r}$ and $B \in \mathbb{R}^{r \times d_{\text{in}}}$, with rank $r \ll \min(d_{\text{out}}, d_{\text{in}})$. The forward pass becomes $Wx + ABx$, where only A and B are learned while W is frozen [39]. This formulation drastically reduces the number of trainable parameters from $d_{\text{out}} \times d_{\text{in}}$ to $r(d_{\text{out}} + d_{\text{in}})$, which can be less than 1% of the original count for typical rank choices such as $r = 4$ or 8 [40]. Moreover, by scaling the LoRA update with a learnable or fixed coefficient α/r, the impact of the low-rank perturbation can be controlled, allowing fine-grained tuning of model behavior [41]. One of the elegant aspects of LoRA is that it integrates seamlessly into existing architectures without altering their inference-time footprint [42]. During deployment, the low-rank matrices A and B can be fused into the original weight matrix W as a single update, effectively collapsing the model back into its original shape [43]. This makes LoRA especially appealing for latency-sensitive applications, as it does not increase the model's depth, width, or memory access patterns [44]. Furthermore, LoRA has been successfully applied to key components of transformer architectures, including the query, key, and value projection matrices in attention blocks, as well as feed-forward layers. Since the self-attention mechanism is sensitive to small perturbations in these weights, fine-tuning via low-rank updates proves to be highly expressive, enabling the model to specialize to new tasks with minimal additional computation. Variants and extensions of LoRA have emerged rapidly, reflecting the method's foundational role in the PEFT ecosystem. For example, LoHa (Low-rank Hadamard Product Atten-

tion) modifies the way attention weights are decomposed, introducing structured element-wise interactions that increase representational capacity without inflating parameter count [45]. Another extension, Q-LoRA, introduces quantization into the fine-tuning pipeline. Q-LoRA freezes the full-precision pre-trained model and quantizes it (e.g., to 4-bit or 8-bit precision) during inference and fine-tuning, while still learning the LoRA weights in full precision [46]. This hybrid strategy dramatically reduces memory consumption during training, making it possible to fine-tune 65B+ models on a single A100 GPU, thus democratizing access to frontier models. The theoretical grounding of LoRA-type methods lies in the observation that many large weight updates during training are actually redundant or highly correlated across neurons. Empirical studies of weight changes in pre-trained models have revealed that the dominant directions of adaptation can be captured by a few principal components, suggesting an inherently low intrinsic dimensionality of the optimization trajectory. By aligning with these low-dimensional subspaces, LoRA not only reduces the number of parameters but also constrains the optimization landscape, potentially improving generalization and reducing overfitting in low-data regimes. Moreover, the composability of LoRA weights allows different tasks to be represented by distinct low-rank matrices, which can be stored, shared, and swapped independently [47]. This property is particularly beneficial in multi-task learning or federated learning settings, where bandwidth and storage constraints make full model replication infeasible. In practical implementations, LoRA typically requires only minor architectural changes and can be enabled via libraries such as HuggingFace PEFT or PEFT-compatible implementations in OpenDelta or Parameter Efficient LLMs. The simplicity of the design belies its powerful implications: LoRA enables fast and memory-efficient adaptation of massive language and vision models to diverse downstream tasks such as machine translation, question answering, image captioning, and medical report generation [48]. Furthermore, LoRA is model-agnostic—it can be applied not only to encoder-decoder models like T5 but also to decoder-only models like GPT, autoregressive image models like MaskGIT, and even hybrid architectures like Flamingo or BLIP. This generality, combined with its practical benefits, has led to widespread adoption in both academia and industry, with LoRA modules now frequently included in public model releases and system pipelines. In summary, low-rank adaptation methods represent a cornerstone of modern parameter-efficient fine-tuning. Their blend of theoretical elegance, practical efficiency, and empirical success makes them a default choice for adapting large pre-trained models to specialized tasks under resource constraints [49]. As models continue to grow and the need for scalable deployment increases, we expect further innovations in this space, including dynamic low-rank adaptation, task-conditioned decompositions, and integration with retrieval-augmented or instruction-based paradigms [50]. LoRA and its variants

not only offer a solution to the practical bottlenecks of full fine-tuning but also open up new research directions in understanding the low-dimensional structure of knowledge transfer in large-scale models.

6 Prompt-Based Fine-Tuning

Prompt-based fine-tuning methods represent a conceptually distinct class of PEFT approaches that reformulate the adaptation problem by shifting the trainable components entirely to the input space [51]. Rather than modifying internal weights or inserting architectural augmentations, prompt-based techniques prepend or infuse learnable vectors—referred to as soft prompts—into the model's inputs or internal activations [52]. This idea draws inspiration from the success of prompt engineering in natural language processing, where cleverly crafted textual prompts can significantly influence the outputs of large language models (LLMs) without any gradient updates. Prompt tuning elevates this idea to a trainable paradigm, wherein the prompts themselves are parameterized as continuous vectors optimized to guide the model toward desired task behavior. Importantly, the pre-trained model remains completely frozen, making this a particularly appealing method for scenarios requiring extreme parameter efficiency and strict preservation of the base model's capabilities [53]. In its simplest instantiation, soft prompt tuning introduces a sequence of trainable embeddings $\boldsymbol{p} = [p_1, p_2, \ldots, p_m]$, where each $p_i \in \mathbb{R}^d$ and d is the model's embedding dimension [54]. These prompts are prepended to the tokenized input sequence, forming an augmented input $[p_1, \ldots, p_m, x_1, \ldots, x_n]$, where x_i are the token embeddings of the original input. The model processes this extended sequence as if it were a regular input, leveraging its existing language modeling capabilities to generate outputs conditioned on the prompt. Since the prompts live in the same vector space as token embeddings, they are initialized either randomly or from existing token vectors and are optimized using standard gradient descent on a task-specific objective. The number of trainable parameters is thus reduced to $m \times d$, often only a few thousand parameters for reasonable values of m (e.g., 20–100) [55]. Despite this extreme reduction in parameter count, prompt tuning has been shown to match or even outperform full fine-tuning on certain tasks, especially when using very large models (e.g., GPT-3 or T5-XXL), where the pre-trained capacity compensates for the simplicity of the learned adaptation. Prefix tuning expands on this concept by inserting learnable vectors not at the input level, but into the key and value projections of attention layers throughout the transformer. For each transformer layer ℓ, learnable prefix vectors $\boldsymbol{k}^{(\ell)}, \boldsymbol{v}^{(\ell)} \in \mathbb{R}^{m \times d}$ are concatenated to the original key and value sequences computed from the input. Specifically, if $K^{(\ell)}$ and $V^{(\ell)}$ are the standard key and

value matrices derived from the input at layer ℓ, the modified attention becomes:

$$\text{Attention}(Q^{(\ell)}, [\boldsymbol{k}^{(\ell)}; K^{(\ell)}], [\boldsymbol{v}^{(\ell)}; V^{(\ell)}]), \tag{8}$$

where $Q^{(\ell)}$ is the query matrix and $[\,;\,]$ denotes concatenation along the sequence dimension. This allows the model to "attend" to the soft prompts during every layer of processing, giving them deeper influence over the model's internal representations. Unlike prompt tuning, which acts solely on the input embeddings, prefix tuning injects control at every transformer block, yielding greater flexibility and expressivity. Moreover, since the prompt vectors are independent of the input, they function similarly to learned control signals, guiding the model to act in task-specific ways while leaving the original attention mechanism untouched. A more recent and powerful evolution of this approach is P-Tuning v2, which unifies prompt tuning and prefix tuning into a single framework and scales well to large models and diverse tasks. It learns hierarchical prompt structures that are injected at multiple layers, potentially with layer-specific transformations [56]. The design emphasizes parameter-sharing and regularization to maintain generalization, while still achieving strong task-specific adaptation [57]. The success of P-Tuning v2 has been demonstrated across the GLUE and SuperGLUE benchmarks, as well as multilingual and cross-domain applications, underscoring the versatility of prompt-based PEFT methods. Prompt-based fine-tuning methods are also naturally aligned with the causal nature of decoder-only models like GPT, which process input in an autoregressive fashion. Because these models condition each token on previous ones, the injected prompt has a persistent influence across the entire forward pass [58]. As such, prompt tuning is particularly effective in tasks like text generation, summarization, and few-shot inference [59]. Moreover, since the frozen model's vocabulary and positional encoding remain intact, prompt tuning preserves linguistic priors encoded during pre-training, thereby ensuring high performance even with minimal downstream supervision. Importantly, the optimization landscape of prompt-based methods differs significantly from that of traditional weight-based fine-tuning: the model must "learn to use" the prompts effectively through its pre-trained mappings, leading to challenges such as prompt drift, task interference, and instability in low-data regimes. Various regularization techniques, such as prompt dropout, orthogonality constraints, and initialization from real token embeddings, have been proposed to mitigate these issues. Beyond language models, prompt-based PEFT techniques have also found success in vision and multimodal domains. In vision-language models like CLIP, prompts can be textual or visual, and tuning them allows for flexible zero-shot and few-shot classification [60]. Visual prompt tuning, for instance, learns a small set of image tokens or spatial perturbations appended to input images processed by vision transformers [61]. These visual prompts can be shared across tasks or specialized

for each, enabling adaptation without modifying the base ViT encoder [62]. Multimodal prompt tuning extends these ideas to cross-attention layers, allowing a unified adapter to influence how visual and textual modalities are fused. These methods maintain the promise of minimal trainable overhead while enabling effective transfer to new tasks such as VQA, image-text retrieval, and captioning [63]. In conclusion, prompt-based fine-tuning represents a compelling trade-off between efficiency and performance [64]. By leveraging the power of learned embeddings and careful insertion points within transformer architectures, these methods enable large pre-trained models to be adapted to new tasks with a fraction of the parameters and no alteration to the model weights. Their compatibility with frozen backbones, versatility across modalities, and potential for plug-and-play inference make them an indispensable tool in the PEFT toolkit. As foundation models continue to grow in scale and ubiquity, the role of prompt-based adaptation strategies is likely to expand further, potentially evolving toward dynamic, context-conditioned prompts, memory-augmented tuning, and hybrid models that blend discrete and continuous prompt components for maximal expressiveness.

7 BitFit and Sparse Fine-Tuning Methods

BitFit and sparse fine-tuning methods offer a minimalistic yet surprisingly effective approach to parameter-efficient adaptation of large pre-trained models. These techniques lie at the extreme end of the PEFT spectrum, proposing that even fewer parameters—sometimes fewer than 0.1% of the total model—need to be updated to achieve meaningful performance on downstream tasks. Unlike adapter-based or low-rank methods that introduce auxiliary modules, or prompt-based approaches that manipulate inputs or activations, BitFit constrains fine-tuning to a small, fixed subset of the original model parameters—often just the bias terms—while sparse methods identify and update a dynamically selected subset of the model weights [65]. The fundamental insight is that large pre-trained models already encode rich and flexible representations, and that minimal task-specific perturbations, strategically placed, may be sufficient to steer model behavior toward a desired target function [66]. BitFit, short for "bias-only fine-tuning," was introduced as an empirical hypothesis: that updating only the bias vectors $\{b^{(\ell)}\}$ in each linear or affine transformation suffices for high performance on many tasks, particularly classification and regression [67]. Let each transformer layer consist of weight matrices $W^{(\ell)}$ and biases $b^{(\ell)}$, with the transformation $f^{(\ell)}(x) = W^{(\ell)}x + b^{(\ell)}$. BitFit proposes to freeze all weights $W^{(\ell)}$ and only optimize the bias vectors $b^{(\ell)}$, leading to a dramatic reduction in trainable parameters. Since the number of bias parameters is on the order of $O(dL)$, where d is the model's hidden size and L is the number of layers, this approach achieves an efficiency unmatched by more

expressive methods. Surprisingly, despite this simplicity, BitFit achieves competitive results with full fine-tuning on many GLUE benchmark tasks, indicating that bias terms alone possess enough flexibility to capture the marginal perturbations required for task-specific adaptation. The mechanism underlying this effectiveness may lie in the fact that biases shift the feature distribution in each layer, subtly altering the activation flow across the network in a globally consistent way [68]. Sparse fine-tuning generalizes this idea by selecting and updating a sparse subset of the model weights, rather than restricting updates to a fixed component such as bias. A sparse mask $\mathcal{M}$ is defined over the weight tensor W, where only entries $W_{i,j}$ for which $\mathcal{M}_{i,j} = 1$ are allowed to be updated during training [69]. The challenge is to identify an optimal mask that captures the most task-relevant parameters under a strict parameter budget [70]. Methods such as DiffPrune, SparseGPT, and Dynamic Sparse Training (DST) propose various criteria for mask selection, ranging from gradient-based heuristics (e.g., update entries with the largest gradient magnitude) to structural or importance-based metrics. Mathematically, the goal is to solve:

$$\min_{\Delta W} \mathcal{L}(f(x; W + \Delta W)) \quad \text{s.t.} \quad \|\Delta W\|_0 \leq k, \tag{9}$$

where $\|\cdot\|_0$ denotes the ℓ_0-norm (counting nonzero entries), and k is the sparsity budget [71]. Since this problem is combinatorially hard, approximate solutions involve pruning strategies, reparameterizations, or sparse learning schedules that gradually reveal or freeze weights over training [72]. Sparse fine-tuning can be layerwise, groupwise, or dynamic, depending on whether sparsity is enforced uniformly across the model or adaptively discovered during training. A notable advantage is that sparsity constraints can be combined with other PEFT strategies, such as LoRA or adapter tuning, to further compress the adaptation space [73]. The sparse tuning paradigm is particularly attractive in resource-constrained or edge deployment scenarios, where every kilobyte of memory or floating-point operation must be justified. Updating only 0.05% of the model—carefully chosen—can outperform naive low-rank adaptations or even full fine-tuning under small data regimes. Moreover, sparsity brings interpretability benefits: understanding which weights matter most for a given task can offer insights into the model's internal structure and inductive biases [74]. In practice, however, sparse fine-tuning suffers from certain limitations. Sparse gradients can lead to optimization instability, and the overhead of mask management may offset memory gains unless implemented efficiently. Recent work has addressed these challenges via sparse tensor libraries and block-sparse GPU kernels that enable high-throughput training on sparsified models. Hybrid approaches have also emerged. For example, DynaSparse or Sparse-LoRA apply LoRA-style low-rank updates only to sparse subspaces of the weight matrices, combining the best of both worlds [75]. Other methods

use lottery ticket-style rewinding, where initializations from pre-training are used to re-initialize sparse subnetworks that are then fine-tuned, ensuring better generalization [76]. BitFit has also been extended into hierarchical variants, where both bias and layer norm parameters are adapted, further enhancing expressiveness while preserving parameter economy. This results in a parameter budget that scales linearly with model depth rather than width, ideal for very deep architectures like BERT-110M or ViT-B [77]. The appeal of BitFit and sparse fine-tuning lies not only in their parameter savings but also in their theoretical implications. They challenge the common assumption that task adaptation requires expressive, high-dimensional parameter changes [78]. Instead, they suggest that the latent manifold of useful solutions is low-dimensional and possibly sparsely connected to the pre-trained weight space. This perspective motivates further exploration into neural reparameterization, efficient subspace optimization, and biologically inspired learning where synaptic changes are sparse and highly localized [79]. In summary, BitFit and sparse tuning push the limits of minimalism in PEFT [80]. They demonstrate that surprisingly strong performance can be obtained with a vanishingly small number of trainable parameters, provided those parameters are appropriately chosen. While they may not match the full flexibility of adapter- or LoRA-based methods, their computational simplicity and interpretability make them valuable tools for practitioners [81]. As hardware evolves to support fine-grained sparsity and as theoretical understanding deepens, these techniques may become the foundation for ultra-lightweight and personalized model deployments at scale.

8 Evaluation and Benchmarking of PEFT Methods

A comprehensive and rigorous evaluation framework is essential for understanding the relative effectiveness, robustness, and transferability of parameter-efficient fine-tuning (PEFT) methods. Since PEFT operates under strict parameter constraints while aiming to approximate or match the performance of full fine-tuning, standard evaluation metrics alone—such as accuracy or F1 score—are often insufficient to capture the nuanced trade-offs involved. Therefore, benchmarking PEFT approaches requires multidimensional assessment criteria, encompassing not only task performance but also adaptation cost, parameter count, computational overhead, convergence stability, and generalization behavior [82]. In addition, the choice of datasets, model sizes, and experimental protocols critically influences conclusions about the comparative utility of PEFT techniques. A well-designed evaluation must isolate variables, quantify trade-offs, and support reproducibility

to guide principled adoption in practical deployments. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$ denote a set of downstream tasks and let f_θ be a pre-trained model with parameters θ [83]. Under PEFT, we introduce a small trainable parameter subset $\phi \subset \theta$ or $\phi \cap \theta = \emptyset$, yielding a new model $f_{\theta,\phi}$. The evaluation goal is to measure the task-specific loss $\mathcal{L}_{T_k}(f_{\theta,\phi})$ for each $T_k \in \mathcal{T}$, while controlling for $\|\phi\|_0$, the number of trainable parameters. A basic metric is the performance gap $\Delta_k = \mathcal{L}_{T_k}(f_{\theta,\phi}) - \mathcal{L}_{T_k}(f_{\theta^*})$, where θ^* denotes the fully fine-tuned model [84]. Averaging over tasks yields the aggregate PEFT gap:

$$\Delta_{\text{avg}} = \frac{1}{K} \sum_{k=1}^{K} \Delta_k, \tag{10}$$

which captures how much performance is lost due to parameter constraints [85]. However, this scalar metric must be contextualized by the relative parameter efficiency $\eta = \frac{\|\phi\|_0}{\|\theta\|_0}$ and training cost $\mathcal{C}(\phi)$ (e.g., FLOPs, wall-clock time), leading to efficiency-adjusted scores such as:

$$\text{EffScore} = \frac{1 - \Delta_{\text{avg}}}{\eta \cdot \mathcal{C}(\phi)}. \tag{11}$$

Such normalized efficiency scores allow direct comparison across diverse methods and hardware platforms, reflecting real-world constraints more accurately than accuracy metrics alone. Benchmarks commonly used for evaluating PEFT include standard NLP tasks such as GLUE, SuperGLUE, and XTREME, covering a range of syntactic, semantic, and multilingual understanding tasks. For vision models, datasets such as ImageNet, CIFAR-100, and VTAB serve as baselines, while multimodal evaluation involves benchmarks like VQAv2, MSCOCO, and OK-VQA. Each benchmark has unique characteristics—task diversity, class imbalance, domain shift—that challenge different aspects of PEFT. For instance, few-shot and low-resource settings are particularly informative for assessing prompt-tuning methods, which often excel with minimal data [86]. In contrast, adapter and LoRA-based methods may outperform in moderate to high-data regimes due to their greater expressivity [87]. Another critical dimension of benchmarking is robustness [88]. Robustness to input perturbations, adversarial attacks, and distribution shifts remains underexplored in the PEFT context. Since many PEFT methods freeze most of the model, they may inherit the brittle generalization characteristics of the pre-trained model without the compensatory flexibility afforded by full fine-tuning. To evaluate this, one may compute robustness margins:

$$\text{RM}_\epsilon = \mathbb{E}_{x \sim \mathcal{D}} \left[\mathcal{L}(f_{\theta,\phi}(x)) - \mathcal{L}(f_{\theta,\phi}(x + \delta)) \right], \quad \text{s.t. } \|\delta\| < \epsilon, \tag{12}$$

which quantify the sensitivity of the PEFT model under ℓ_p-bounded perturbations. Robustness-aware metrics like RM_ϵ or adversarial accuracy under PGD or

FGSM attacks are vital for deployment in high-stakes settings [89]. The choice of base model also affects benchmarking outcomes. Many PEFT studies rely on fixed-size encoders such as BERT-base, RoBERTa-large, or ViT-B/16 to ensure comparability, but performance gaps and efficiency gains often widen as model size increases [90]. It has been empirically observed that soft prompt tuning and BitFit, in particular, scale better with model size than smaller PEFT modules such as adapters, which may underfit on large-scale backbones. Therefore, scaling curves—performance vs. model size—should be included in evaluation to understand how PEFT methods behave asymptotically. In addition, ablation studies and sensitivity analyses are critical for understanding the behavior of PEFT under hyperparameter variation [91]. For example, prompt length, rank of LoRA matrices, layer insertion depth for adapters, and sparsity ratio for sparse tuning can dramatically affect downstream performance [92]. A complete evaluation report should include grid or random search results, as well as robustness to initialization variance (e.g., across seeds) [93]. Moreover, it is increasingly standard to perform few-shot, in-domain, and cross-domain evaluations in a unified framework to assess generalization [94]. Finally, reproducibility and transparency are vital [95]. Many PEFT results are sensitive to subtle factors such as learning rate warm-up, layer norm treatment, or positional encoding schemes. Therefore, benchmark repositories such as PEFTBench, AdapterHub, and HuggingFace's PEFT library play a crucial role in standardizing evaluation protocols and promoting fair comparison. Ideally, each reported result should include model card metadata detailing training conditions, parameter counts, GPU hours, and dataset licenses to promote sustainable and ethical research practices [96]. In conclusion, evaluating PEFT methods is an inherently multi-faceted task that requires going beyond raw accuracy to encompass efficiency, robustness, scalability, and reproducibility [97]. As PEFT becomes a cornerstone of practical fine-tuning for foundation models, future benchmarks must evolve to reflect emerging applications, support multimodal scenarios, and enable dynamic model selection under hard constraints. Only with such comprehensive evaluation can we truly understand the cost-performance frontier of parameter-efficient adaptation.

9 Applications of PEFT in Natural Language, Vision, and Multimodal Domains

Parameter-efficient fine-tuning (PEFT) techniques have demonstrated wide applicability across a spectrum of domains where deploying large pre-trained models under limited compute, storage, or privacy constraints is crucial [98]. The growing diversity of PEFT approaches—ranging from low-rank updates and adapters

to prompt-based tuning and sparsity-aware methods—has enabled practitioners to tailor adaptation strategies to a variety of application-specific desiderata, such as rapid domain adaptation, low-latency inference, personalization, and continual learning [99]. As foundation models become increasingly ubiquitous across industry and research, the operational feasibility of adapting these models with minimal overhead has shifted PEFT from a theoretical optimization problem to a central practical concern. This section surveys the deployment of PEFT methods in major application domains, highlighting domain-specific considerations, performance trade-offs, and case studies of real-world adoption [100]. In natural language processing (NLP), PEFT methods have been instrumental in adapting large language models (LLMs) such as GPT-3, BERT, and T5 to downstream tasks like sentiment classification, question answering, information retrieval, and dialogue generation. One major application driver is low-resource adaptation: tasks with few labeled examples or domain shifts (e.g., biomedical, legal, or social media corpora) benefit immensely from tuning methods that require few parameters and exhibit strong inductive biases. For instance, prompt tuning has found significant success in zero- and few-shot learning settings, particularly when combined with instruction tuning or retrieval-augmented models [101]. Adapter-based methods, on the other hand, have proven particularly useful in multi-task and multilingual NLP, where the modularity of adapters enables flexible parameter reuse [102]. Consider a multilingual BERT system augmented with task-specific adapters for each language-pair translation task; the overall model size remains nearly constant while supporting dozens of translation directions [103]. More recently, LoRA and QLoRA have enabled parameter-efficient instruction-following tuning of massive LLMs (e.g., LLaMA-65B) using consumer-grade hardware, democratizing access to powerful generative capabilities and accelerating development cycles for chatbots, summarization tools, and creative writing assistants [104]. In computer vision (CV), the application of PEFT is relatively newer but growing rapidly, especially with the rise of Vision Transformers (ViTs) and contrastive pretraining paradigms such as CLIP. Unlike CNNs, ViTs are naturally more amenable to PEFT strategies due to their modular layer structures and high redundancy in self-attention blocks [105]. Adapter-based methods, particularly those placed after attention and MLP layers, have been effectively used for image classification, object detection, and semantic segmentation tasks [106]. For example, in transfer learning from ImageNet to medical imaging datasets like ChestX-ray14, fine-tuning only a small number of adapters significantly outperforms linear probing while preserving computational tractability. LoRA has also been successfully applied to tune visual transformers for domain-specific applications such as satellite imagery and autonomous driving, where full model retraining is prohibitive [107]. Moreover, sparse and BitFit-like methods in vision remain underexplored, but initial results indicate that even small

shifts in biases or structured subspace updates can yield surprisingly high returns in cross-domain settings with minimal data. The multimodal domain represents a particularly exciting frontier for PEFT, as it necessitates the adaptation of multi-branch architectures with cross-modal attention and modality-specific encoders. Prominent examples include models like Flamingo, GIT, and CLIP, which jointly process vision and language to perform tasks such as visual question answering (VQA), captioning, and referring expression comprehension. Here, the benefits of PEFT are twofold: parameter savings facilitate scalability to new modalities (e.g., audio, video, sensor data), while modularity enables fine-grained control over which modalities are adapted, frozen, or jointly tuned. In practice, a common strategy is to fine-tune only the cross-modal fusion layers via low-rank or adapter modules, while freezing the modality-specific encoders (e.g., a ViT and a BERT) [108]. This avoids catastrophic forgetting and enables rapid deployment to novel multimodal tasks with limited supervision. Additionally, prompt-based approaches can be extended to multimodal prompting (e.g., visual prompts or joint embeddings), enabling zero-shot transfer from text-only pretraining to image-text alignment tasks [109]. Personalization is another emergent area where PEFT shines. For language models deployed in conversational agents, email assistants, or recommendation engines, personalization requires user-specific tuning without retraining the entire model for each individual. LoRA and BitFit allow creating low-footprint user adapters or embedding offsets that encode personal preferences, writing style, or task history. These personalized modules can be stored and switched dynamically, enabling real-time adaptation while preserving user privacy and global model integrity. Similarly, in vision-based applications such as AR/VR or gaze tracking, user calibration modules can be implemented via PEFT techniques to fine-tune model behavior with only a few calibration samples [110]. Continual learning and domain adaptation offer further fertile ground for PEFT. In these regimes, the model must adapt to a stream of changing tasks or environments without catastrophic forgetting of earlier knowledge [111]. Adapter-based architectures are particularly useful here, as they allow new tasks to be added as plug-in modules while preserving the core model [112]. Some methods use sparsity constraints or orthogonal projection schemes (e.g., Orthogonal Weight Modification) to avoid interference between tasks [113]. In robotics, for instance, continual adaptation to new environments or object types can be achieved by stacking LoRA modules per environment, preserving task-specific behavior without resetting the model or maintaining multiple full copies. In summary, the applications of PEFT methods are broad, impactful, and rapidly expanding across natural language, vision, and multimodal domains [114]. They provide practical solutions to longstanding challenges in efficient deployment, personalization, and robust adaptation, making them indispensable in modern AI systems. As PEFT methods continue to evolve

and mature, their integration into end-to-end pipelines—from pretraining to deployment—will become standard practice, driving innovation across academia and industry [115].

10 Conclusion and Future Directions

Parameter-efficient fine-tuning (PEFT) has emerged as a transformative paradigm for adapting large-scale pre-trained models across diverse tasks, domains, and modalities. By selectively updating a small subset of parameters or introducing compact learnable modules, PEFT techniques offer a compelling trade-off between performance, efficiency, and scalability [116]. This survey has provided a comprehensive overview of the PEFT landscape—covering soft and hard prompt tuning, adapter modules, low-rank adaptation techniques such as LoRA and QLoRA, sparse and BitFit-based methods, as well as a unified treatment of their mathematical foundations, empirical behaviors, and practical implications. Across language, vision, and multimodal tasks, PEFT has consistently demonstrated the ability to match or even surpass full fine-tuning baselines under dramatically reduced compute and memory footprints [117]. The generality, modularity, and cost-effectiveness of these methods have enabled widespread adoption in both academic research and industrial deployment. Despite these advances, many challenges remain unresolved, offering rich directions for future research. One of the most fundamental open questions concerns the theoretical understanding of why PEFT works so well [118]. What properties of the pre-trained parameter landscape allow for such high-fidelity task adaptation in low-dimensional or sparse subspaces [119]? Are there universal inductive biases that certain PEFT strategies implicitly exploit, such as alignment with principal directions of the Fisher information matrix or layerwise curvature profiles? Answering these questions may require new theoretical frameworks that combine insights from information geometry, representation learning, and low-rank matrix theory. Closely related is the need for better interpretability: understanding which parameters, features, or modules are responsible for successful adaptation could guide the design of even more efficient methods. A second important direction lies in the design of PEFT strategies that go beyond static architectures. Most current approaches operate by inserting or optimizing fixed modules (e.g., a rank-r matrix in LoRA, or a prompt embedding of length l). However, dynamic and data-dependent architectures—where the size, position, or content of the adaptation modules evolves during training—remain underexplored. Such methods could adaptively allocate parameter budgets based on gradient saliency, token importance, or activation entropy, leading to models that self-regularize their parameter usage based on task complexity. Meta-learning, reinforcement learning, and neural architecture search may offer powerful tools for

automatically discovering efficient PEFT configurations. The interaction between PEFT and other machine learning paradigms also invites further study [120]. For example, how can PEFT be effectively integrated with continual learning to enable seamless, lifelong adaptation [121]? Can PEFT enhance the robustness and fairness of foundation models by enabling targeted updates without global retraining? In federated learning and edge AI, PEFT offers a natural fit due to its low communication cost and high modularity—but methods must be developed to handle heterogeneous data, privacy constraints, and noisy updates [122]. Similarly, in the context of generative models such as diffusion models or instruction-tuned LLMs, PEFT techniques tailored for generation-specific components (e.g., decoder-only attention blocks, cross-attention layers, or denoising steps) remain underdeveloped. Such extensions may be key to scaling generative AI safely and sustainably. Another frontier is the unification of PEFT across modalities. While individual methods have been applied successfully in NLP and CV, unified architectures that support efficient adaptation across text, vision, audio, and other sensor modalities remain rare. Building such systems requires designing PEFT modules that are modality-agnostic, interoperable, and tunable at multiple granularity levels [123]. This could enable truly generalist models that can be deployed on any device, customized for any user, and updated continuously in response to new tasks. Integrating this with multimodal alignment, cross-modal prompts, or shared tokenization schemes may give rise to a new generation of highly adaptable and efficient AI agents. Lastly, benchmarking and standardization must keep pace with innovation [124]. The community would benefit from a rigorous, modular PEFT benchmark suite that includes not just accuracy metrics, but also measures of training time, memory usage, inference latency, robustness, and environmental impact. Such benchmarks should support reproducibility across hardware platforms and include realistic deployment scenarios such as few-shot learning, personalization, and domain shift [125]. Community-driven repositories, shared baselines, and transparent evaluation protocols will be essential to ensure that progress in PEFT is cumulative, grounded, and ethically aligned [126].

In conclusion, PEFT has redefined how we approach fine-tuning in the era of large foundation models. Its flexibility, efficiency, and versatility make it not just an optimization technique, but a foundational design principle for the next generation of adaptable AI systems. As models grow ever larger and are deployed across increasingly dynamic, distributed, and personalized environments, PEFT will play a central role in making powerful AI both accessible and sustainable. The field stands at a promising inflection point—one where theoretical insights, methodological innovations, and real-world applications can converge to unlock the full potential of parameter-efficient adaptation.

References

[1] Bruce XB Yu, Jianlong Chang, Lingbo Liu, Qi Tian, and Chang Wen Chen. Towards a unified view on visual parameter-efficient transfer learning. *arXiv preprint arXiv:2210.00788*, 2022.

[2] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[3] Yun-Yun Tsai, Chengzhi Mao, and Junfeng Yang. Convolutional visual prompt for robust visual perception. *Advances in Neural Information Processing Systems*, 2023.

[4] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[5] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on lora finetuning dynamics. *arXiv preprint arXiv:2406.08447*, 2024.

[6] Suyi Li, Hanfeng Lu, Tianyuan Wu, Minchen Yu, Qizhen Weng, Xusheng Chen, Yizhou Shan, Binhang Yuan, and Wei Wang. Caraserve: Cpu-assisted and rank-aware lora serving for generative llm inference. *arXiv preprint arXiv:2401.11240*, 2024.

[7] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Jiazhang Lian, Qiang Zhang, and Huajun Chen. Ontoprotein: Protein pretraining with gene ontology embedding. *ArXiv*, abs/2201.11147, 2022.

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2024.

[9] Hengyu Zhang. Sinklora: Enhanced efficiency and chat capabilities for long-context large language models. *arXiv preprint arXiv.2406.05678*, 2023.

[10] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63:1872 – 1897, 2020.

[11] Joshua Belofsky. Token-level adaptation of lora adapters for downstream task generalization. In *6th Artificial Intelligence and Cloud Computing Conference*, pages 168–172, 2023.

[12] Yifan Chen, Devamanyu Hazarika, Mahdi Namazifar, Yang Liu, Di Jin, and Dilek Hakkani-Tur. Empowering parameter-efficient transfer learning by recognizing the kernel structure in self-attention. *arXiv preprint arXiv:2205.03720*, 2022.

[13] Jiazheng Xing, Mengmeng Wang, Xiaojun Hou, Guang Dai, Jingdong Wang, and Yong Liu. Multimodal adaptation of clip for few-shot action recognition. *arXiv preprint arXiv:2308.01532*, 2023.

[14] Yassine Zniyed, Thanh Phuong Nguyen, et al. Efficient tensor decomposition-based filter pruning. *Neural Networks*, 178:106393, 2024.

[15] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *ArXiv*, abs/2206.06522, 2022.

[16] Baohao Liao and Christof Monz. Apiq: Finetuning of 2-bit quantized large language model. *arXiv preprint arXiv:2402.05147*, 2024.

[17] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *ArXiv*, abs/2106.10199, 2021.

[18] Yixuan Ren, Yang Zhou, Jimei Yang, Jing Shi, Difan Liu, Feng Liu, Mingi Kwon, and Abhinav Shrivastava. Customize-a-video: One-shot motion customization of text-to-video diffusion models. *ECCV*, 2024.

[19] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[21] Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. A mathematical perspective on transformers. *arXiv preprint arXiv.2312.10794*, 2023.

[22] Szymon Tworkowski, Konrad Staniszewski, Mikolaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Milos. Focused transformer: Contrastive training for context scaling. In *Advances in Neural Information Processing Systems*, 2023.

[23] Yassine Zniyed, Thanh Phuong Nguyen, et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[24] Shu Yang, Muhammad Asif Ali, Cheng-Long Wang, Lijie Hu, and Di Wang. Moral: Moe augmented lora for llms' lifelong learning. *arXiv preprint arXiv.2402.11260*, 2024.

[25] Yi Xin, Junlong Du, Qiang Wang, Zhiwen Lin, and Ke Yan. Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[26] Jiacheng Zhu, Kristjan H. Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv.2402.16842*, 2024.

[27] Xiao-Yang Liu, Rongyi Zhu, Daochen Zha, Jiechao Gao, Shan Zhong, Matt White, and Meikang Qiu. Differentially private low-rank adaptation of large language model using federated learning. *ACM Transactions on Management Information Systems*, 2023.

[28] Huanjin Yao, Wenhao Wu, and Zhiheng Li. Side4video: Spatial-temporal side network for memory-efficient image-to-video transfer learning. *arXiv preprint arXiv:2311.15769*, 2023.

[29] Sloke Shrestha, Asvin Venkataramanan, et al. Style transfer to calvin and hobbes comics using stable diffusion. *arXiv preprint arXiv:2312.03993*, 2023.

[30] Shuying Zhang, Jing Zhang, Hui Zhang, and Li Zhuo. Rastformer: region-aware spatiotemporal transformer for visual homogenization recognition in short videos. *Neural Computing and Applications*, 2024.

[31] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 2020.

[32] Shaoxu Li. Diffstyler: Diffusion-based localized image style transfer. *arXiv preprint arXiv:2403.18461*, 2024.

[33] Colin Raffel, Noam M. 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. *ArXiv*, abs/1910.10683, 2019.

[34] Adam Gaier and David R Ha. Weight agnostic neural networks. In *Neural Information Processing Systems*, 2019.

[35] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

[36] Qidong Huang, Xiaoyi Dong, Dongdong Chen, Weiming Zhang, Feifei Wang, Gang Hua, and Nenghai Yu. Diversity-aware meta visual prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[37] Y Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[38] Sanghyeon Kim, Hyunmo Yang, Yunghyun Kim, Youngjoon Hong, and Eunbyung Park. Hydra: Multi-head low-rank adaptation for parameter efficient fine-tuning. *Neural Networks*, page 106414, 2024.

[39] Gongye Liu, Menghan Xia, Yong Zhang, Haoxin Chen, Jinbo Xing, Xintao Wang, Yujiu Yang, and Ying Shan. Stylecrafter: Enhancing stylized text-to-video generation with style adapter. *arXiv preprint arXiv:2312.00330*, 2023.

[40] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

[41] Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[42] Mohit Sharma, Claudio Fantacci, Yuxiang Zhou, Skanda Koppula, Nicolas Heess, Jon Scholz, and Yusuf Aytar. Lossless adaptation of pretrained vision models for robotic manipulation. *International Conference on Learning Representations*, 2023.

[43] Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. Structure-aware low-rank adaptation for parameter-efficient fine-tuning. *Mathematics*, 11(20):4317, 2023.

[44] You Zhang, Jin Wang, Liang-Chih Yu, Dan Xu, and Xuejie Zhang. Personalized lora for human-centered text understanding. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, pages 19588–19596, 2024.

[45] Jiawei Zheng, Hanghai Hong, Xiaoli Wang, Jingsong Su, Yonggui Liang, and Shikai Wu. Fine-tuning large language models for domain-specific machine translation. *arXiv preprint arXiv:2402.15061*, 2024.

[46] Mang Ye, Xiuwen Fang, Bo Du, Pong C. Yuen, and Dacheng Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):79:1–79:44, 2024.

[47] Mushui Liu, Bozheng Li, and Yunlong Yu. Omniclip: Adapting clip for video recognition with spatial-temporal omni-scale feature learning. *arXiv preprint arXiv:2408.06158*, 2024.

[48] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[49] Shyam Marjit, Harshit Singh, Nityanand Mathur, Sayak Paul, Chia-Mu Yu, and Pin-Yu Chen. Diffusekrona: A parameter efficient fine-tuning method for personalized diffusion model. *arXiv preprint arXiv:2402.17412*, 2024.

[50] Mu Cai, Haotian Liu, Siva Karthik Mustikovela, Gregory P Meyer, Yuning Chai, Dennis Park, and Yong Jae Lee. Vip-llava: Making large multimodal models understand arbitrary visual prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[51] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, 2023.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[53] Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. Seed-x: Multimodal models with

unified multi-granularity comprehension and generation. *arXiv preprint arXiv:2404.14396*, 2024.

[54] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[55] Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. Lora land: 310 fine-tuned llms that rival gpt-4, A technical report. *arXiv preprint arXiv.2405.00732*, 2024.

[56] Yitao Liu, Chenxin An, and Xipeng Qiu. $$\cal{Y}$$-tuning: an efficient tuning paradigm for large-scale pre-trained models via label representation learning. *Frontiers Comput. Sci.*, 18(4), 2024.

[57] Han Lin, Jaemin Cho, Abhay Zala, and Mohit Bansal. Ctrl-adapter: An efficient and versatile framework for adapting diverse controls to any diffusion model. *arXiv preprint arXiv:2404.09967*, 2024.

[58] Michael E. Sander, Pierre Ablin, Mathieu Blondel, and Gabriel Peyré. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, pages 3515–3530, 2022.

[59] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.*, 5(3):220–235, 2023.

[60] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5217–5227, 2021.

[61] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[62] Weihuang Liu, Xi Shen, Chi-Man Pun, and Xiaodong Cun. Explicit visual prompting for low-level structure segmentations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[63] Mengjie Zhao, Tao Lin, Martin Jaggi, and Hinrich Schütze. Masking as an efficient alternative to finetuning for pretrained language models. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[64] Weijia Feng, Lingting Zhu, and Lequan Yu. Cheap lunch for medical image segmentation by fine-tuning SAM on few exemplars. *arXiv preprint arXiv:2308.14133*, 2023.

[65] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

[66] Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. Implicit style-content separation using b-lora. *arXiv preprint arXiv:2403.14572*, 2024.

[67] Bohdan M. Pavlyshenko. Financial news analytics using fine-tuned llama 2 GPT model. *arXiv preprint arXiv:2308.13032*, 2023.

[68] Zhihao Wen, Jie Zhang, and Yuan Fang. Sibo: A simple booster for parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.11896*, 2024.

[69] Yuanzhao Zhai, Han Zhang, Yu Lei, Yue Yu, Kele Xu, Dawei Feng, Bo Ding, and Huaimin Wang. Uncertainty-penalized reinforcement learning from human feedback with diverse reward lora ensembles. *arXiv preprint arXiv:2401.00243*, 2024.

[70] Dongshuo Yin, Xueting Han, Bin Li, Hao Feng, and Jing Bai. Parameter-efficient is not sufficient: Exploring parameter, memory, and time efficient adapter tuning for dense predictions. *arXiv preprint arXiv:2306.09729*, 2023.

[71] Pengcheng He, Baolin Peng, Liyang Lu, Song Wang, Jie Mei, Yang Liu, Ruochen Xu, Hany Hassan Awadalla, Yu Shi, Chenguang Zhu, et al. Z-code++: A pre-trained language model optimized for abstractive summarization. *arXiv preprint arXiv:2208.09770*, 2022.

[72] Xin Zhou, Dingkang Liang, Wei Xu, Xingkui Zhu, Yihan Xu, Zhikang Zou, and Xiang Bai. Dynamic adapter meets prompt tuning: Parameter-efficient transfer learning for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[73] Yaohua Zha, Jinpeng Wang, Tao Dai, Bin Chen, Zhi Wang, and Shu-Tao Xia. Instance-aware dynamic prompt tuning for pre-trained point cloud models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[74] 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 *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799, 2019.

[75] Shamil Ayupov and Nadezhda Chirkova. Parameter-efficient finetuning of transformers for source code. *arXiv preprint arXiv:2212.05901*, 2022.

[76] Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations. *arXiv preprint arXiv.2306.14870*, 2023.

[77] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

[78] Jie Qin, Jie Wu, Pengxiang Yan, Ming Li, Ren Yuxi, Xuefeng Xiao, Yitong Wang, Rui Wang, Shilei Wen, Xin Pan, et al. Freeseg: Unified, universal and open-vocabulary image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19446–19455, 2023.

[79] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Differentially private bias-term only fine-tuning of foundation models. *Advances in Neural Information Processing Systems*, 2022.

[80] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, 2023.

[81] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *ArXiv*, abs/1905.01067, 2019.

[82] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663, 2016.

[83] Aochuan Chen, Yuguang Yao, Pin-Yu Chen, Yihua Zhang, and Sijia Liu. Understanding and improving visual prompting: A label-mapping perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19133–19143, 2023.

[84] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[85] Augustin Toma, Patrick R. Lawler, Jimmy Ba, Rahul G. Krishnan, Barry B. Rubin, and Bo Wang. Clinical camel: An open-source expert-level medical language model with dialogue-based knowledge encoding. *arXiv preprint arXiv:2305.12031*, 2023.

[86] Yang Sui, Miao Yin, Yu Gong, Jinqi Xiao, Huy Phan, and Bo Yuan. ELRT: efficient low-rank training for compact convolutional neural networks. *arXiv preprint arXiv.2401.10341*, 2024.

[87] Shaoxiang Chen, Zequn Jie, and Lin Ma. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms. *arXiv preprint arXiv.2401.16160*, 2024.

[88] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. SLoRA: Federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*, 2023.

[89] Bohao Peng, Jian Wang, Yuechen Zhang, Wenbo Li, Ming-Chang Yang, and Jiaya Jia. Controlnext: Powerful and efficient control for image and video generation. *arXiv preprint arXiv:2408.06070*, 2024.

[90] Yudi Zhang, Qi Xu, and Lei Zhang. Dragtex: Generative point-based texture editing on 3d mesh. *arXiv preprint arXiv:2403.02217*, 2024.

[91] Duarte M. Alves, Nuno Miguel Guerreiro, João Alves, José Pombal, Ricardo Rei, José Guilherme Camargo de Souza, Pierre Colombo, and André F. T. Martins. Steering large language models for machine translation with finetuning and in-context learning. In *Findings of the Association for Computational Linguistics*, pages 11127–11148, 2023.

[92] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[93] Wele Gedara Chaminda Bandara and Vishal M Patel. Attention prompt tuning: Parameter-efficient adaptation of pre-trained models for spatiotemporal modeling. *arXiv preprint arXiv:2403.06978*, 2024.

[94] Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. Emoticon smoothed language models for twitter sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1678–1684, 2012.

[95] Yubin Wang, Xinyang Jiang, De Cheng, Dongsheng Li, and Cairong Zhao. Actprompt: In-domain feature adaptation via action cues for video temporal grounding. *arXiv preprint arXiv:2408.06622*, 2024.

[96] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

[97] 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.

[98] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *ArXiv*, abs/2210.04284, 2022.

[99] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022.

[100] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32, 2019.

[101] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge J. Belongie, Bharath Hariharan, and Ser Nam Lim. Visual prompt tuning. *ArXiv*, abs/2203.12119, 2022.

[102] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[103] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings*, 2020.

[104] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *Advances in Neural Information Processing Systems*, 2022.

[105] Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *ArXiv*, abs/2102.01386, 2021.

[106] Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter efficient multi-task model fusion with partial linearization. *arXiv preprint arXiv.2310.04742*, 2023.

[107] Jiao Sun, Deqing Fu, Yushi Hu, Su Wang, Royi Rassin, Da-Cheng Juan, Dana Alon, Charles Herrmann, Sjoerd van Steenkiste, Ranjay Krishna, et al. Dreamsync: Aligning text-to-image generation with image understanding feedback. In *Synthetic Data for Computer Vision Workshop@ CVPR 2024*, 2023.

[108] Zhehuai Chen, He Huang, Andrei Andrusenko, Oleksii Hrinchuk, Krishna C. Puvvada, Jason Li, Subhankar Ghosh, Jagadeesh Balam, and Boris Ginsburg. SALM: speech-augmented language model with in-context learning for speech recognition and translation. *arXiv preprint arXiv:2310.09424*, 2023.

[109] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[110] Ning Ding, Yujia Qin, Guang Yang, Fu Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Haitao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juan Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *ArXiv*, abs/2203.06904, 2022.

[111] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023.

[112] Kaidong Zhang and Dong Liu. Customized segment anything model for medical image segmentation. *arXiv preprint arXiv:2304.13785*, 2023.

[113] Panlong Wu, Kangshuo Li, Ting Wang, and Fangxin Wang. FedMS: Federated learning with mixture of sparsely activated foundations models. *arXiv preprint arXiv:2312.15926*, 2023.

[114] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[115] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[116] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[117] Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*, 2023.

[118] Feihu Jin, Yin Liu, and Ying Tan. Derivative-free optimization for low-rank adaptation in large language models. *arXiv preprint arXiv:2403.01754*, 2024.

[119] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021.

[120] Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*, 2024.

[121] Lingmin Ran, Xiaodong Cun, Jia-Wei Liu, Rui Zhao, Song Zijie, Xintao Wang, Jussi Keppo, and Mike Zheng Shou. X-adapter: Adding universal compatibility of plugins for upgraded diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[122] Yitao Liu, Chenxin An, and Xipeng Qiu. Y-tuning: An efficient tuning paradigm for large-scale pre-trained models via label representation learning. *Frontiers of Computer Science*, 18(4):184320, 2024.

[123] Yiwei Ma, Yijun Fan, Jiayi Ji, Haowei Wang, Xiaoshuai Sun, Guannan Jiang, Annan Shu, and Rongrong Ji. X-dreamer: Creating high-quality 3d content by bridging the domain gap between text-to-2d and text-to-3d generation. *arXiv preprint arXiv:2312.00085*, 2023.

[124] 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.

[125] Zhouxia Wang, Xintao Wang, Liangbin Xie, Zhongang Qi, Ying Shan, Wenping Wang, and Ping Luo. Styleadapter: A single-pass lora-free model for stylized image generation. *arXiv preprint arXiv:2309.01770*, 2023.

[126] Jiayi Guo, Xingqian Xu, Yifan Pu, Zanlin Ni, Chaofei Wang, Manushree Vasu, Shiji Song, Gao Huang, and Humphrey Shi. Smooth diffusion: Crafting smooth latent spaces in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.