Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey

Zeyu Han[1], Chao Gao[2], Jinyang Liu[1], Jeff (Jun) Zhang[3], and Sai Qian Zhang[4]

[1]*Northeastern University*   [2]*University of California, Riverside*   [3]*Arizona State University*
[4]*New York University*

{han.zeyu,liu.jinyan}@northeastern.edu, cgao037@ucr.edu, jeffzhang@asu.edu, sai.zhang@nyu.edu

Abstract—Large models represent a groundbreaking advancement in multiple application fields, enabling remarkable achievements across various tasks. However, their unprecedented scale comes with significant computational costs. These models, often consisting of billions of parameters, require vast amounts of computational resources for execution. Especially, the expansive scale and computational demands pose considerable challenges when customizing them for particular downstream tasks, particularly over the hardware platforms constrained by computational capabilities.

Parameter Efficient Fine-Tuning (PEFT) provides a practical solution by efficiently adapt the large models over the various downstream tasks. In particular, PEFT refers to the process of adjusting the parameters of a pre-trained large models to adapt it to a specific task or domain while minimizing the number of additional parameters introduced or computational resources required. This approach is particularly important when dealing with large-scale language models with high parameter counts, as fine-tuning these models from scratch can be computationally expensive and resource-intensive, posing considerable challenges in the supporting system platform design.

In this survey, we present comprehensive studies of various PEFT algorithms, examining their performance and computational overhead. Moreover, we provide an overview of applications developed using different PEFT algorithms and discuss common techniques employed to mitigate computation costs for PEFT. In addition to the algorithmic perspective, we overview various real-world system designs to investigate the implementation costs associated with different PEFT algorithms. This survey serves as an indispensable resource for researchers aiming to understand both the PEFT algorithm and its system implementation, offering detailed insights into recent advancements and practical applications.

Index Terms—Large Language Model, Parameter-Efficient Fine-tuning, Computer System, Distributed System.

I. INTRODUCTION

Large Models (LMs) have recently captured considerable public interest. Their ability to understand context and nuances enables them to proficiently handle diverse tasks across multiple domains, including natural language processing (NLP), computer vision (CV), etc. In the field of NLP, Large Language Models (LLMs) have achieved significant advancements across various tasks including text generation [1], [2], translation [3], [4], personalized chat-bots [5], [6], [7], and summarization [8], demonstrating remarkable proficiency.

Earlier studies [1] has suggested that LLMs exhibit high levels of generalization, enabling them to apply their acquired knowledge to new tasks not included in their original training. This capability is commonly known as *zero-shot learning*. Nevertheless, fine-tuning remains essential to further enhance LLMs for optimal performance on new user datasets and tasks.

Due to its scale, a widely adopted strategy for fine-tuning LLMs involves adjusting a limited number of LLM parameters while keeping the remainder unchanged. This technique, termed *Parameter-Efficient-Fine-Tuning* (PEFT), involves selectively adjusting a small proportion of their parameters, while keeping the rest unaltered. Furthermore, the application of PEFT extends beyond the realm of NLP and quickly attracts interest in the CV community for handling fine-tuning vision models with large parameters, such as Vision Transformers (ViT) and diffusion models, as well as disciplinary models such as vision-language models.

In this survey, we systematically review and categorize recent advancements in PEFT algorithms as well as the system implementation costs associated with various PEFT algorithms across diverse scenarios. Figure 1 presents the overview content for this survey. In section II, we present some fundamental concepts for LLM and PEFT, including computational flow for LLM, basic knowledge of PEFT, and commonly used datasets and tasks.

We categorized all types of PEFT algorithms in Section III according to their computational flow. In Section III-A, we introduce additive algorithms that either introduce additional weight parameters or modify activations. For algorithms that exclusively require fine-tuning with existing parameters, they fall under the category of selective approaches, and their introduction can be found in Section III-B. In Section III-C, we explore reparameterized PEFT, which constructs a (low-dimensional) reparameterization of original model parameters for training while transforms the weights back to maintain the inference speed. Additionally, there exist algorithms that combine the above techniques, and we have classified these as hybrid approaches, elaborating on them in Section III-D. We also investigate strategies for further reducing the computational complexity of different PEFT algorithms, including KV-cache management, pruning, quantization, and memory optimization, in Section IV.

In Section V, we expand the scope of this survey beyond the computational perspective to involve various potential application scenarios. We explore innovations that applying PEFT techniques to different model architecture, including LLMs
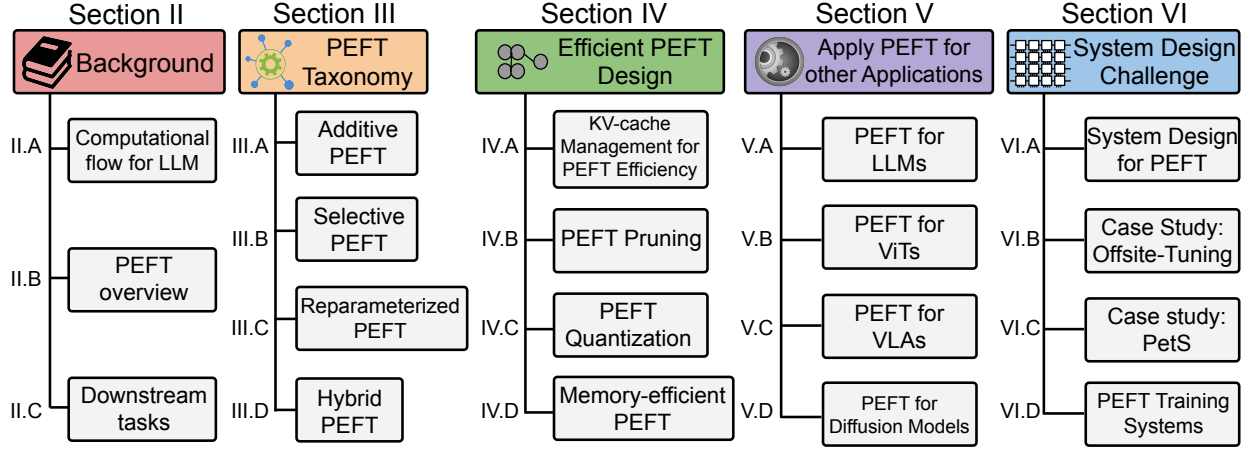
Fig. 1: A content overview covered in the survey.

(Section V-A), Vision Transformer (Section V-B), Vision-Language alignment models (Section V-C), and Diffusion models (Section V-D), for varied downstream tasks, underscoring PEFT's versatility and applicability in a range of scenarios.

In Section VI, we explores the system design challenge for PEFT methods. The discussion includes three advanced system solutions for practical PEFT deployment: distributed tuning (Section VI-B), PEFT query serving (Section VI-C), and concurrent PEFT tuning (Section VI-D).

In the last Section VII, we summarize our survey and propose several potential future directions from both algorithm and system perspectives, hoping to provide valuable insights for further research and development in the field.

II. BACKGROUND

In this section, we first discussed the computation flow of LLM, including its fundamental components, computational complexity, and the flow of computations it involves as a case study. We then provide a brief overview of different PEFT algorithms in section II-B.

A. Computation flow for LLaMA

In order to gain a deeper understanding of LLM and other Transformer-based models, we employ LLaMA-7B, a cutting-edge open-source LLM model, to scrutinize the architecture of LLM as well as Transformer. As shown in Figure 2 (a), LLaMA consists of three major components: an embedding block, a stack of decoder blocks and a head block which consists of linear and softmax layer. The embedding layer's primary role is to transform unstructured textual information, into chunks of discrete numerical vectors (*tokens*) to facilitate subsequent processing. The embedded tokens are then delivered to the decoder layers for further processing. Each LLaMA decoder is composed of two fundamental components: Multi-head Self-Attention (MSA) and Feedforward Network (FFN). In the MSA module, each of the tokens will be clustered by an attention map obtained by a dot production between two linear mappings of the input tokens. Then the grouped tokens will be further processed by a Feedforward Neural network.

Additionally, Root Mean Square Layer Normalization (RMSNorm) [9] is adopted in LLaMA as a replacement for Layer Normalization to ensure efficient training.

LLM distinguishes itself from other deep neural network (DNN) models such as convolutional neural networks (CNN) in two significant ways. Firstly, LLM exhibits an inherent autoregressive nature, necessitating multiple iterations to complete the generation task. Moreover, LLM incorporates an attention mechanism, a component with computational complexity that scales quadratically with the length of the inputs. On the other hand, the inherent computation characteristic of LLM lies in the attention blocks inside each decoder layer. Figure 2 (c) depicts the high-level overview of the computation flow in the attention block.

During the inference process, each decoder takes a 4-dimensional tensor $x \in \mathbb{R}^{b \times l \times d}$ as the input tokens. The input tokens are first multiplied with three weight matrices W_Q, W_K, and W_V, producing the output referred to as query(Q), key(K) and value(V). Given the MSA module's inability to recognize positional data and the inherent autoregressive nature of LLMs, the query and key will undergo a process using Rotary Positional Embedding [10] (RoPE, denoted as $R(.)$ in Eq 1) to encode the position information. Subsequently, the key and value will be combined with prior tokens.

After the positional embedding, the intermediate activation will then undergo a series of multiplication, softmax, and residual addition to generate MSA output as described in Eq 9. To be noted here, d_k in the equation refers to the number of feature dimensions in the multi-head attention mechanism.

$$Q, K, V = R(W_q x), R(W_k x), W_v x \qquad (1)$$

$$SA(x) = Softmax(\frac{QK^T}{\sqrt{d_{head}}})V \qquad (2)$$

$$MSA(x) = [SA_1(x); SA_2(x); \ldots; SA_k(x)]W_o \qquad (3)$$

The SA output will then be forwarded to the FFN blocks for further processing. The FFN block will have another three
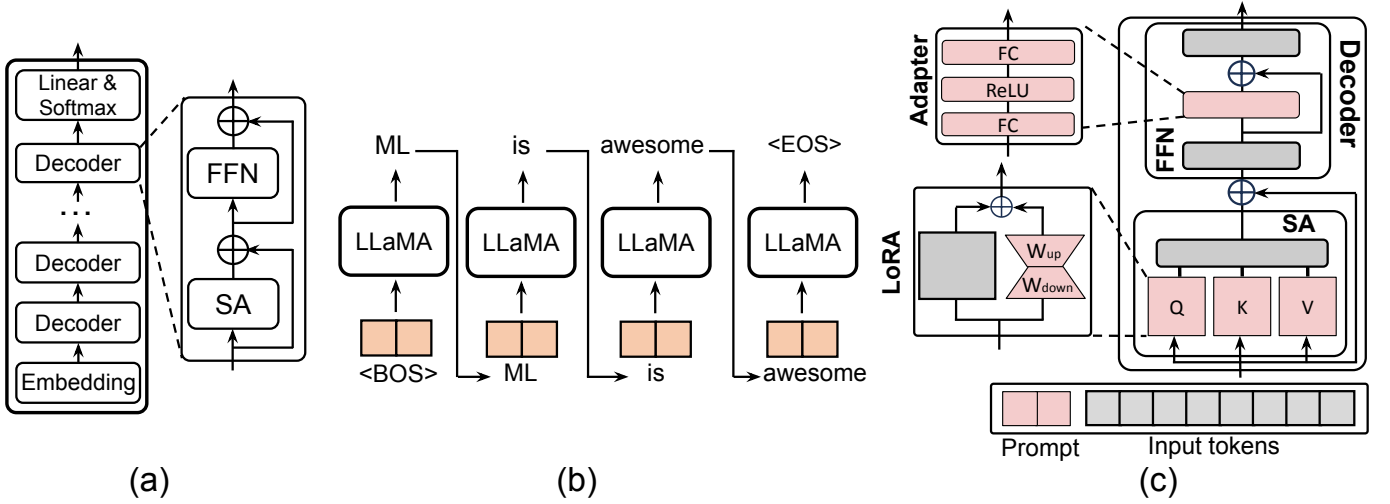
Fig. 2: (a) LLaMA architecture. (b) LLaMA auto-regressive pattern. (c) Three common PEFT operations. All the learnable components are highlighted in red, while the frozen components are highlighted in grey. LoRA is applied on all the Query, Key, and Value blocks. The adapter targets the FFN module. Soft-Prompt focused on tuning the input activation of each decoder. We only show one decoder for illustration simplicity.

matrices W_{up}, W_{down}, and W_{gate} and the computation can be illustrated by:

$$FFN_{LLaMa}(x) = W_{up}(SiLU(W_{gate}x) \odot (W_{down}x)) + x, \tag{4}$$

where x denotes the input of the FFN layer, and $SiLU$ is the nonlinear function used in LLaMA. In the original Transformer, the FFN block can be demonstrated by:

$$FFN_{Transfomer}(x) = W_{up}(ReLU(W_{down}x)) + x. \tag{5}$$

The output of the last decoder layer will be sent to a linear layer, which then generates a probability distribution spanning the complete *vocabulary* to predict the next token in the sequence. The produced token will then be concatenated with the previous tokens and used as the input for the next round of processing. This generating process repeats in an auto-regressive manner until a full sequence of tokens, referred to as *completion*, is produced (Figure 2 (b)). For training, the computation flow is similar to that for inference, except that the generated sentences are directly compared to the ground truth output and generate the training loss. Gradients will then be computed across the LLM weights to minimize this training loss.

To analyze the computation cost and memory overhead in LLM, we also set a series of parameters used in later section III. Table I shows the parameter size and computation dimension in the LLaMA-7B model as a starting example.

LLM models generate tokens (words) one for each round, depicted in Fig 2, based on the previous prompt (input) and previously generated sequence. This process will be repeated until the model outputs hits and termination token. To accelerate the inference process in LLM models, people take the strategy of storing the previous Keys and Values in Key-Value cache (KV-cache), so they don't need to recalculate them for each new token. Mathematically, we can represent the total decoders' kv-cache memory cost in equation 6. In

the equation, l and b are the context length and batch size and L refers to the number of layers. The d_{head} is the head dimension and n_{head} is the number of heads.

$$Size = L \times 2 \times b \times l \times d_{head} \times n_{head} \tag{6}$$

B. Overview on Parameter Efficient Fine Tuning

Fine-tuning remains essential to enhance LLM performance on unseen user datasets and tasks. With the size of the model growing (e.g. 1.5B in GPT-2 to 175B in GPT-3), standard full fine-tuning paradigm requires thousands of GPU work in parallel, which is highly inefficient and unsustainable. A type of algorithm has been raised namely Parameter-efficient fine-tuning (PEFT) which aims to tune minimal parameters to achieve better performance over full tuning on downstream tasks.

In parallel developments, large-scale pre-trained models in vision and multimodal domains have also demonstrated their effective representational learning capabilities, enabling adaptation from large datasets to smaller ones or across various data modalities through fine-tuning. Consequently, this capability has made PEFT increasingly attractive to the wider research community.

We categorized the PEFT algorithms into **additive, selective, reparameterized, and hybrid** fine-tuning based on their operations. As Figure 3 depicts, three major **additive fine-tuning** algorithms are normally used: (1) Adapter; (2) Soft Prompt; (3) Others. They differ from each other in terms of the different additional tunable modules or parameters. **Selective fine-tuning**, on the other hand, doesn't require any additional parameters, it selects a small subset of parameters from the backbone model and only makes them tunable while keeping the majority of parameters untouched during fine-tuning on downstream tasks. We categorized selective fine-tuning based on the grouping of chosen parameters: (1) Unstructural Masking; (2) Structural Masking. Reparametrization

Operation	Weights Symbol	Weights Dimension	Input Tensor Dimension	Complexity
Eq. 1	W_Q, W_K, W_V	$d \times k \times \frac{d}{k}$	$b \times l \times d$	$O(l)$
Eq. 2	-	-	$b \times l \times 3 \times k \times \frac{d}{k}$	$O(l^2)$
Eq. 3	W_o	$d \times d$	$b \times l \times d$	$O(l)$
Eq. 4	$W_{up}, W_{down}, W_{gate}$	$d \times 4d$	$b \times l \times d$ OR $l \times b \times 4d$	$O(l)$

TABLE I: Configuration parameters and computation operation for LLaMA-7B architecture

represents transforming model parameters between two equivalent forms. Specifically, **reparametrized fine-tuning** introduces additional low-rank trainable parameters during training, which are then integrated with the original model for inference. This approach is categorized into two main strategies: (1) Low-rank Decomposition, and (2) LoRA Derivatives. **Hybrid fine-tuning** explores the design spaces of different PEFT methods and combines their advantages.

C. Downstream Tasks for LLM Evaluation

Two types of tasks have been widely used for LLM evaluation, the first type is the General Language Understanding Evaluation (GLUE) [11] benchmark, which integrates nine sentence or sentence-pair language understanding tasks (CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, RTE, and WNLI), chosen for their diversity in dataset sizes, text genres, and difficulty levels, and is based on established existing datasets. It also includes a diagnostic dataset specifically designed to evaluate and analyze model performance across various linguistic phenomena inherent in natural language. Additionally, it features a public leaderboard to track performance on the benchmark and a dashboard to visualize model performance on the diagnostic set.

The other type of dataset that has been used in recent LLM papers is common sense reasoning which integrated into our study caters to a variety of research facets: (1) *OpenBookQA* [12] is curated to foster research in advanced question-answering, delving into a profound understanding of both the subject matter and the language in which it is articulated. (2) *PIQA* [13] primarily emphasizes everyday scenarios, demonstrating a predilection for unconventional solutions. (3) *Social IQA* [14] emerges as a novel question-answering benchmark tailored for gauging social common-sense intelligence. (4) *HellaSwag* [15] serves as a dataset, the essence of which is to ascertain the capability of machines in aptly concluding sentences. (5) *BoolQ* [16] is a dataset dedicated to question-answering, particularly for binary responses (yes/no queries). (6) *WinoGrande* [17] is introduced as a fresh compilation, encompassing a substantial 44,000 problems. (7) *ARC-easy* [18] presents itself as a novel dataset constituting genuine grade-school level multiple-choice science questions, designed to invigorate research in intricate question-answering. (8) *ARC-challenges* [18], distinctively, encompasses solely those questions that were inaccurately addressed by both a retrieval-based algorithm and a word co-occurrence algorithm.

Image recognition is the primary benchmark and application for vision models, exemplified by benchmarks such as fine-grained visual categorization (FGVC) and visual task adaptation benchmark (VTAB). Beyond image classification, video action recognition is another key application area, involving datasets like Kinetics-400 [19], SSv2 [20], and HMDB51 [21]. Additionally, PEFT has been utilized for dense prediction tasks, using datasets like MSCOCO [22], ADE20K [23], and PASCAL VOC [24].

III. PEFT TAXONOMY

The PEFT strategies can be broadly classified into four categories: **additive PEFT** (Section III-A), which modifies the model architecture by injecting new trainable modules or parameters; **selective PEFT** (Section III-B), which makes a subset of parameters trainable during fine-tuning; **reparameterized PEFT** (Section III-C), which constructs a (low-dimensional) reparameterization of the original model parameters for training, then equivalently transforms it back for inference; and **hybrid PEFT** (Section III-D), which combines advantages from different PEFT methods to build a unified PEFT model. A overview of different types of PEFT algorithms is depicted in Figure 4.

A. Additive PEFT

Standard full fine-tuning entails substantial computational expenses and also could potentially harm the model's generalization ability. To mitigate this problem, a widely employed approach is to maintain the pre-trained backbone unchanged and introduce only a minimal number of trainable parameters that are strategically positioned within the model architecture. While fine-tuning for a specific downstream task, only the weights of these additional modules or parameters are updated, which results in a substantial reduction in storage, memory, and computational resource requirements. Due to their characteristic of adding parameters, these techniques can be termed as *Additive Tuning*, as shown in Figure 4 (a). Next, we discuss several popular Additive PEFT algorithms.

1) Adapters: Adapter approaches involve the insertion of small adapter layers within Transformer blocks. Typically, an adapter layer consists of a down-projection matrix $W_{\text{down}} \in \mathbb{R}^{r \times d}$, followed by a non-linear activation function $\sigma(\cdot)$, and an up-projection matrix $W_{\text{up}} \in \mathbb{R}^{d \times r}$. In this context, d represents the dimension of the hidden layer, and r serves as the bottleneck dimension, which is a hyperparameter used in configuring the adapters. Denote h_{in} as the input to the adapter, the computation within the adapter module (with residual) can be summarized as follows:
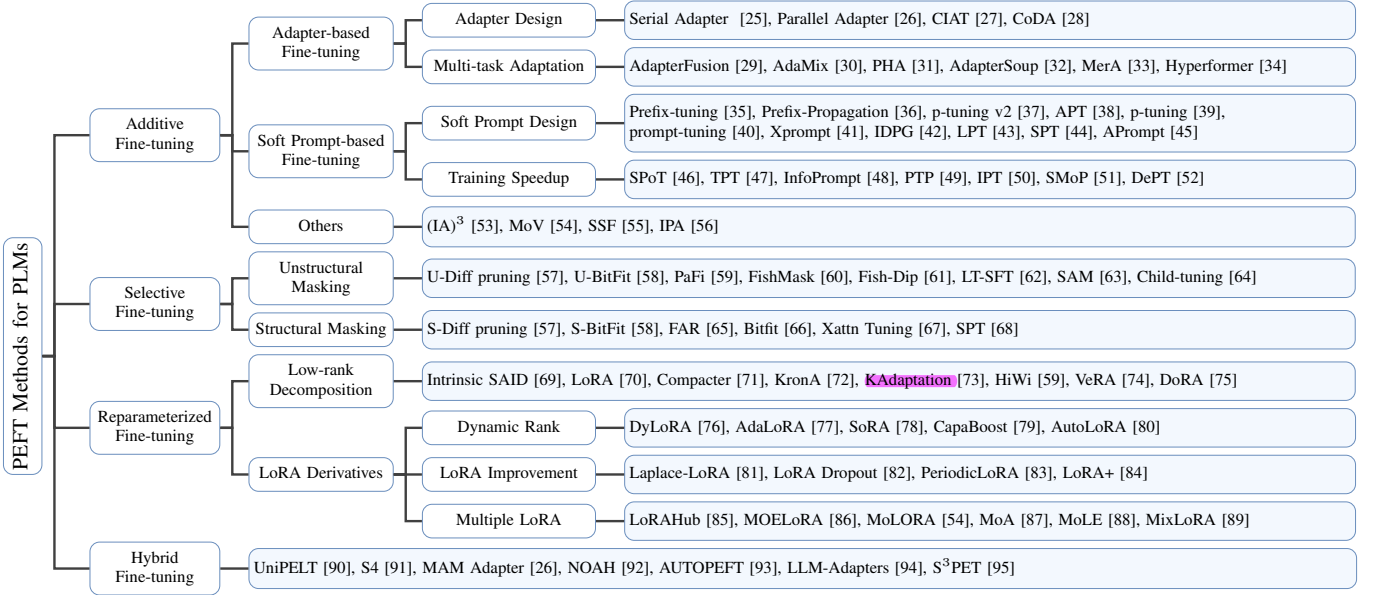
PEFT Methods for PLMs

- **Additive Fine-tuning**
  - **Adapter-based Fine-tuning**
    - **Adapter Design** — Serial Adapter [25], Parallel Adapter [26], CIAT [27], CoDA [28]
    - **Multi-task Adaptation** — AdapterFusion [29], AdaMix [30], PHA [31], AdapterSoup [32], MerA [33], Hyperformer [34]
  - **Soft Prompt-based Fine-tuning**
    - **Soft Prompt Design** — Prefix-tuning [35], Prefix-Propagation [36], p-tuning v2 [37], APT [38], p-tuning [39], prompt-tuning [40], Xprompt [41], IDPG [42], LPT [43], SPT [44], APrompt [45]
    - **Training Speedup** — SPoT [46], TPT [47], InfoPrompt [48], PTP [49], IPT [50], SMoP [51], DePT [52]
  - **Others** — (IA)^3 [53], MoV [54], SSF [55], IPA [56]
- **Selective Fine-tuning**
  - **Unstructural Masking** — U-Diff pruning [57], U-BitFit [58], PaFi [59], FishMask [60], Fish-Dip [61], LT-SFT [62], SAM [63], Child-tuning [64]
  - **Structural Masking** — S-Diff pruning [57], S-BitFit [58], FAR [65], Bitfit [66], Xattn Tuning [67], SPT [68]
- **Reparameterized Fine-tuning**
  - **Low-rank Decomposition** — Intrinsic SAID [69], LoRA [70], Compacter [71], KronA [72], KAdaptation [73], HiWi [59], VeRA [74], DoRA [75]
  - **LoRA Derivatives**
    - **Dynamic Rank** — DyLoRA [76], AdaLoRA [77], SoRA [78], CapaBoost [79], AutoLoRA [80]
    - **LoRA Improvement** — Laplace-LoRA [81], LoRA Dropout [82], PeriodicLoRA [83], LoRA+ [84]
    - **Multiple LoRA** — LoRAHub [85], MOELoRA [86], MoLORA [54], MoA [87], MoLE [88], MixLoRA [89]
- **Hybrid Fine-tuning** — UniPELT [90], S4 [91], MAM Adapter [26], NOAH [92], AUTOPEFT [93], LLM-Adapters [94], S^{3}PET [95]

Fig. 3: Taxonomy of Parameter-Efficient Fine-Tuning Methods for Large Models.

(a) Additive PEFT   (b) Selective PEFT   (c) Reparameterization PEFT

Frozen   Learnable
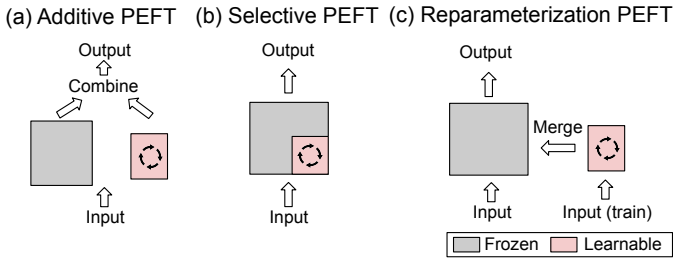
Fig. 4: Different types of PEFT algorithms.

$$Adapter(x) = W_{\text{up}}\sigma(W_{\text{down}}x) + x. \qquad (7)$$

The concept of adapters in the field of NLP was initially introduced by **Serial Adapter** [25] as shown in Figure 5 (a). In their approach, each Transformer block is enhanced by adding two adapter modules, with one positioned after the self-attention layer and the other after the FFN layer, respectively. Subsequent research has aimed to address the additional computational cost associated with adapter layers. A modified framework **AdapterFusion** [29] was proposed, where adapter layers are inserted only after the 'Add & Norm' step following the FFN layer to enhance the computational efficiency. The adapters mentioned above follow a sequential design, placing adapter layers as bottlenecks within Transformer blocks. This approach may potentially reduce the model's parallelism and require a trade-off between inference efficiency and accuracy. In contrast, [26] introduced a **parallel adapter** (PA) approach as depicted in Figure 5 (b), which reorganizes the traditionally sequential adapter layers into a parallel side-network that runs alongside each Transformer sublayer. Similarly, **CIAT** [27], **CoDA** [28] and **KronA** [72] also adopts a parallel adapter design. Except for the parallel design, **CoDA** employs a sparse activation mechanism to improve the inference efficiency as shown in Figure 5 (c).

Specifically, CoDA use a soft top-k selection process that identifies k important tokens in each layer, which will be processed by both the frozen pre-trained Transformer layer and the adapter branch to maintain model accuracy. In contrast, those unimportant tokens are only processed by the adapter branch while skip the heavy pre-trained layer, therefore optimizing for inference efficiency without compromising overall performance.

To enhance the performance and generalization of adapters, various studies have implemented *multi-task learning* strategies, such as **AdapterFusion** [29], **AdaMix** [30], **PHA** [31], **AdapterSoup** [32], **MerA** [33], and **Hyperformer** [34]. **AdapterFusion** keeps all pre-trained adapters in the model and employs a fusion module to merge the multi-task informations. Unlike AdapterFusion, **MerA** merges pretrained adapters into a single one through optimal transport based on weights and activations. This approach avoids introducing any additional trainable parameters, thereby enhancing computational efficiency. **Hyperformer** stores the multi-task information in a shared hypernetwork, which generates task and layer-specific adapter parameters conditioned on task and layer id embeddings. Given a new task, only an additional task embedding needs to be learned, therefore reducing the number of trained parameters.

2) Soft Prompt: Alternatively, prompt tuning presents an additional approach for refining the model to achieve improved performance through fine-tuning. Instead of optimizing discrete token representations through in-context learning, there is a prevailing belief that the continuous embedding space of soft prompts inherently contains more information [96]. Drawing inspiration from this concept, researchers directly append adjustable vectors, referred to as soft prompts, to the start of the input sequence. This can be represented as follows:

$$\mathbf{X}^{(l)} = [\mathbf{s}_1^{(l)}, \ldots, \mathbf{s}_{N_S}^{(l)}, \mathbf{x}_1^{(l)}, \ldots, \mathbf{x}_{N_X}^{(l)}] \qquad (8)$$
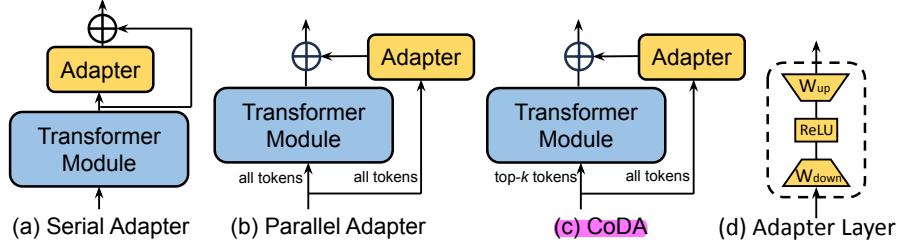
Fig. 5: Illustration of three representative adapter-based fine-tuning algorithms. Blue represents frozen, while yellow represents trainable.

where $\mathbf{X}^{(l)}$ is the sequence of input tokens for layer l, including soft prompt tokens $\mathbf{s}_i^{(l)}$ followed by the original input tokens $\mathbf{x}_i^{(l)}$. N_S is the number of soft prompt tokens, and N_X is the number of original input tokens.

Prefix-tuning [35] introduces learnable vectors that are prepended to keys k and values v across all Transformer layers. To ensure stability during the optimization process, Prefix-tuning adopts a reparameterization strategy, which utilizes an MLP layer to generate these prefix vectors rather than optimizing them directly. After fine-tuning, only the prefix vectors are saved for inference. This technique is adapted and improved in several studies [36], [37], [38]. For instance, **p-tuning v2** [37] removes reparameterization and expands its usage to broader model scales and NLP tasks. **APT** (Adaptive Prefix Tuning) [38] enhances Prefix-tuning by introducing adaptive gate mechanism to control the prefix importance in each layer. Concurrent work **p-tuning** [39] and **prompt-tuning** [40] apply learnable vectors only at the initial word embedding layer rather than all layers to enhance training and inference efficiency. It's important to highlight that prompt-tuning demonstrates its effectiveness primarily in the context of large models, specifically those with over 11 billion parameters [40]. Complementing this, **Xprompt** [41] eliminates the negative prompt tokens through a hierarchical structured pruning, which closes the performance gap at smaller model scales. **The work by [97]** provides some theoretical analysis towards prompt tuning, demonstrating its universality and limitations in limited-depth Transformers. **IDPG** (Instance-Dependent Prompt Generation) [42] improves prompt tuning by generating prompts based on each input sentence with a lightweight prompt generator. In a related approach, **LPT** (Late Prompt Tuning) [43] also leverages a prompt generator to obtain instance-aware prompt. Unlike previous work, LPT adds these prompts only after an intermediate layer, rather than at the initial or all layers. This strategic placement eliminates the gradient calculation below the intermediate layer, thereby significantly accelerating the training speed. Simultaneously, LPT can improve the overall performance due to the shorter backpropagation path preserves more task-related information. Inspired by LPT, **SPT** (Selective Prompt Tuning) [44] delves deeper into the importance of prompt inserting strategies. It introduces a learnable probabilistic gate in each layer to determine whether to use the prompt propagated from the previous layer or inject a newly generated prompt. **APrompt** [45] employs another prompt inserting strategy. In addition to input

prompts inserted at the beginning of the input sequence for each Transformer layer, APrompt also prepends additional learnable prompts to the respective query, key, and value matrices in the self-attention blocks to learn new attention patterns. Besides, APrompt incorporates the learning of a task-specific head.

The concept of soft prompts has been employed for various downstream tasks [98], [99], although their training can be prone to instability and slow convergence. To address this, **SPoT** [46] uses a source prompt learned from one or multiple tasks to initialize prompts for new tasks. Similarly, transfer of soft prompts from one task to initialize another is proposed in **TPT** (transferable prompt tuning) [47], which demonstrates that a better prompt initialization results in a large training convergence speedup. **InfoPrompt** [48] develops two mutual information based loss functions, i.e., *head loss* and *representation loss*, to find better prompt initialization and learn sufficient task-relevant information, thereby also expediting convergence. **PTP** [49] delves into the root causes of training instability. It identifies the steep nature of the loss landscape in conventional prompt tuning, where minor variations in input data can lead to significant loss fluctuations. To mitigate this, PTP introduces perturbation-based regularizers to smooth the loss landscape and consequently stabilize the training process. **DePT** [52] decomposes the soft prompt into a shorter soft prompt with a pair of low-rank matrices, which are optimised with two distinct learning rates. This strategy not only improves the performance but also enhances training and inference efficiency. **SMoP** (Sparse Mixture-of-Prompts) [51] reduce the training and inference cost by utilizing short soft prompts. During training, multiple short soft prompts are trained, each tailored to specific subsets of the dataset. During inference, SMoP integrates a gating mechanism that routes each input instance to an appropriate short prompt. This technique not only increases efficiency in both training and inference stages but also retains performance comparable to those achieved with longer soft prompts. To further cut down the number of soft prompt parameters, **IPT** (Intrinsic Prompt Tuning) [50] identifies an intrinsic task subspace by training an auto-encoder on multiple tasks. Tuning on new tasks then requires adjusting only a few parameters within this subspace, significantly reducing the number of training parameters.

3) Other Additive Methods: Apart from the methods mentioned above, there appears other approaches that strategically incorporate additional parameters during the fine-tuning process. For example, $\mathbf{(IA)}^3$ [53] introduces three learnable
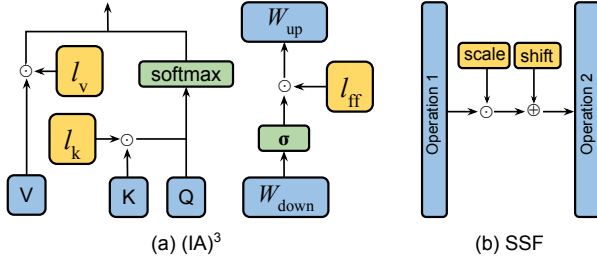
Fig. 6: Illustration of (IA)^3 and SSF. Blue represents frozen, while yellow represents trainable.

rescaling vectors: $l_k \in \mathbb{R}^{d_k}$, $l_v \in \mathbb{R}^{d_v}$, and $l_{ff} \in \mathbb{R}^{d_{ff}}$, to rescale the key, value, and FFN activations, respectively, as depicted in Figure 6 (a). The operations within the self attention block can be described as follows:

$$SA(x) = Softmax\left(\frac{Q(l_k \odot K^T)}{\sqrt{d_{head}}}\right)((l_v \odot V). \quad (9)$$

In FFN, the rescaling can be denoted as:

$$FFN_{Transfomer}(x) = W_{up}(l_{ff} \odot \sigma(W_{down}x)), \quad (10)$$

where $\odot$ is Hadamard product. Furthermore, the scale vectors l_k and l_v can be seamlessly integrated into the weight matrices of A_Q and A_W. This integration effectively eliminates the extra computational costs during inference. A similar technique SSF [55] also performs linear transformation to the model activations, as illustrated in Figure 6 (b). Specifically, after each operation (i.e., MSA, FFN and layer normalization) in the pre-trained model, a SSF-ADA layer is injected, which performs scaling and shifting to the features generated from the operation. During fine-tuning, only those SSF-ADA layers can be updated, while during inference, similar to (IA)^3, these SSF-ADA layers can be merged into model weights, so no additional inference overhead would be incurred. **IPA** (Inference-Time Policy Adapters) [56] offers a novel approach to align LLMs, such as GPT-4, with user-specific requirements without modifying the base model's parameters. This is particularly significant when dealing with models whose parameters are extremely large and often not directly accessible. IPA achieves this by combining (through multiplication and normalization) the output distribution of a base LLM (base policy) with that of a smaller-sized model (adapter policy) during the decoding phase. During training, the policy adapter's parameters are fine-tuned using reinforcement learning, while the base policy's parameters remain fixed. During inference, IPA decodes with the combined distribution of the base model and the trained policy adapter, tailoring it to fulfill specific user-defined criteria.

B. Selective PEFT

Rather than additive PEFT, which increases the model complexity by adding more parameters, selective PEFT fine-tunes a subset of the existing parameters to enhance model performance over downstream tasks, as depicted in Figure 4 (b).
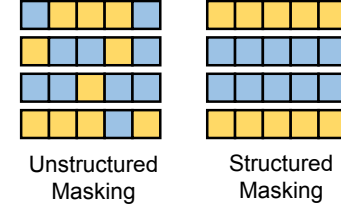

Fig. 7: Illustration of how two parameter masking methods. Blue represents frozen, while yellow represents trainable.

Specifically, given a model with parameters $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$ where each θ_i denotes an individual model parameter and n represents the total count of these parameters, the process of selective PEFT is represented by applying a binary mask $M = \{m_1, m_2, ..., m_n\}$ to these parameters. Each m_i in M is either 0 or 1, indicating whether the corresponding parameter θ_i is selected (1) or not selected (0) for fine-tuning. The updated parameter set θ' after fine-tuning is given by:

$$\theta'_i = \theta_i - \eta \cdot m_i \cdot \frac{\partial \mathcal{L}}{\partial \theta_i} \quad (11)$$

where η represents the learning rate, and $\frac{\partial \mathcal{L}}{\partial \theta_i}$ is the gradient of the loss function with respect to the parameter θ_i. In this formulation, only the parameters that are selected (i.e., $m_i = 1$) are updated during backpropagation.

Diff pruning [57] is a representative work that applies a learnable binary mask to the model weights during fine-tuning. To achieve parameter efficiency, the mask is regularized by a differentiable approximation of the L_0-norm penalty. **PaFi** [59] simply select model parameters with the smallest absolute magnitude as trainable. **FishMask** [60] determines parameter importance using the approximate Fisher information. It then selects the top k parameters based on this information to form the mask M. Similarly, **Fish-Dip** [61] also uses Fisher information to calculate M, but the mask will be re-calculated dynamically in each train period. **LT-SFT** [62] introduces another technique to determines parameter importance inspired by the Lottery Ticket Hypothesis [100], [101], where the the subset of parameters that change the most during an initial fine-tuning stage is selected to form the mask M. **SAM** [63] proposes a second-order approximation method, which approximates the original problem with an analytically solvable optimization function, to help decide the parameter mask. **Child-tuning** [64] proposes two approaches to select a child network during each training iteration, where only the parameters within this child network can be updated.

However, above unstructured parameter masking results in an uneven distribution of non-zero masks and diminished hardware efficiency when implementing PEFT. As shown in Figure 7, the structured mask organize parameter masking in regular patterns, unlike unstructured ones that apply it randomly, thus can enhances computational and hardware efficiency during training. Therefore, various structured selective PEFT techniques have undergone extensive investigation. **Diff pruning** proposes a structured pruning strategy by partitioning the weight parameters into local groups and strategically eliminating them together. Similarly, **FAR** [65]
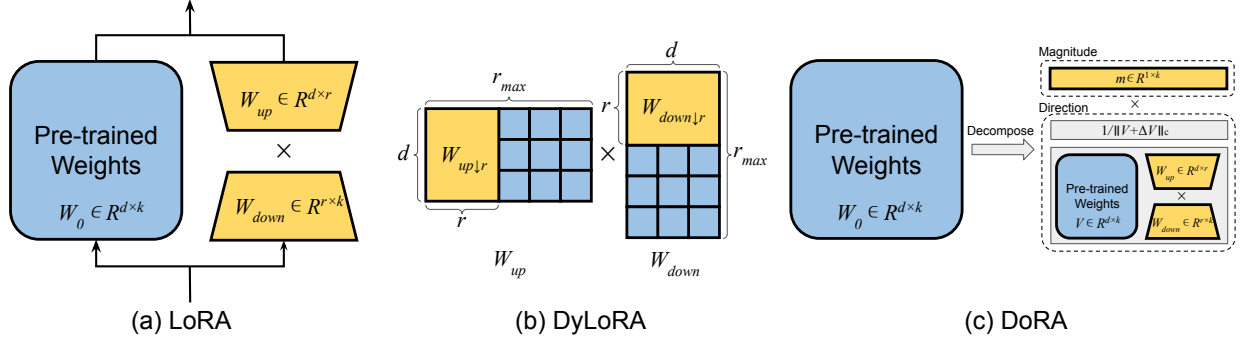
Fig. 8: Illustration of three representative reparameterized PEFT algorithms. Blue represents frozen, while yellow represents trainable.

fine-tunes BERT models by grouping weights of the FFN in Transformer blocks into nodes, then rank and select the learner nodes using L_1 norm. To further reduce the memory access frequency, they also reconfigure the FFN by grouping the learner nodes together. **Bitfit** [66] is proposed to only fine-tunes the bias parameters of each DNN layer, and achieves competitive results for small models. However, this method fails to handle large models. **The work by [58]** applies NAS to Bitfit, where **S-BitFit** keeps the structural nature in Bitfit that restrict NAS algorithm must choose whether $\delta b = 0$ or not for each bias module. Similar to Bitfit that fine-tunes a specific module in Transformer, **Xattn Tuning** [67] fine-tunes only the cross-attention layers. **SPT** (sensitivity-aware visual parameter-efficient fine-tuning) [68] first identifies the sensitive parameters measured by the loss reduction when being tuned. This sensitivity is calculated using a first-order Taylor expansion, derived from a single forward and backward pass before fine-tuning in oneshot. Next, SPT finds the weight matrices whose number of sensitive parameters exceeds a pre-defined threshold, and then applies a selected PEFT technique (e.g., LoRA and Adapter) to these targeted weights to achieve structural tuning.

C. Reparameterized PEFT

Reparameterization stands for equivalently transforming a model's architecture from one to another via transforming its parameters. In the context of PEFT, this often means constructing a low-rank parameterization to achieve the goal of parameter efficiency during training. For inference, the model can be converted to its original weight parameterization, ensuring unchanged inference speed. This procedure is depicted in Figure 4 (c).

Earlier research studies [69] have shown that common pre-trained models exhibit an exceptionally low intrinsic dimensionality. In other words, it is possible to find a low-dimensional reparameterization that is effective for fine-tuning as the entire parameter space. **Intrinsic SAID** [69] is the pioneering work in investigating the intrinsic dimension feature during the fine-tuning of LLMs. However, the most widely recognized reparameterization technique is **LoRA** (Low Rank Adaptation) [70], [102], as shown in Figure 8 (a). For a given pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, LoRA introduces two

trainable weight matrices, $W_{\text{up}} \in \mathbb{R}^{d \times r}$ and $W_{\text{down}} \in \mathbb{R}^{r \times k}$ where the rank $r \ll min(d, k)$, operating in parallel to W_0. Let h_{in} represent the input. Under normal conditions, the output through W_0 is $h_{out} = W_0 h_{in}$. Instead, LoRA modifies this output by introducing an incremental update ΔW that encapsulates task-specific knowledge:

$$h_{out} = W_0 h_{in} + \frac{\alpha}{r} \Delta W h_{in} = W_0 h_{in} + \frac{\alpha}{r} W_{\text{up}} W_{\text{down}} h_{in},$$
(12)

where α denotes a scaling factor. At the onset of training, W_{down} is initialized using a random Gaussian distribution, while W_{up} is initialized to zero, ensuring that ΔW initially holds a value of zero. LoRA is straightforward to implement and has been evaluated on models with up to 175 billion parameters. Fig 8 (c) used a single decoder as an example, the frozen and learnable components are highlighted in grey and red, respectively. Once fine-tuning is complete, LoRA's adaptive weights seamlessly integrate with the pre-trained backbone weights. This integration ensures that LoRA maintains the model's efficiency, adding no extra burden during inference.

In LoRA training, selecting an appropriate rank has always been a challenging issue. To address this, **DyLoRA** [76], as depicted in Figure 8 (b), trains LoRA module on a range of ranks within a predefined training budget, rather than adhering to a single, fixed rank. Specifically, for a given rank range $R = \{r_{\min}, r_{\min}+1, \ldots, r_{\max}\}$, DyLoRA dynamically chooses a rank $r \in R$ at each iteration of the training process. Consequently, the matrices W_{down} and W_{up} are tailored for the selected rank r, resulting in truncated versions $W_{\text{down}\downarrow r} = W_{\text{down}}[1 : r, :]$ and $W_{\text{up}\downarrow r} = W_{\text{up}}[:, 1 : r]$, and the subsequent forward and backward pass during this iteration will be restricted on $W_{\text{down}\downarrow r}$ and $W_{\text{up}\downarrow r}$ instead of W_{down} and W_{up}. With this dynamic and search-free approach, DyLoRA significantly reduces the training time required to find an optimal and fixed LoRA rank for specific tasks. **AdaLoRA** [77] reformulates the ΔW with a singular value decomposition (SVD), denoted as $\Delta W = P \Lambda Q$, where $P \in \mathbb{R}^{d \times r}$ and $Q \in \mathbb{R}^{r \times k}$ are orthometric, Λ is a diagonal matrix containing sigular values $\{\lambda_i\}_{1 \le i \le r}$. All the three weight matrices are made learnable. During training, the singular values are pruned iteratively based on their importance scores, which are constructed from moving average of the magnitude of gradient-weight product.

To ensure the orthogonality between P and Q, i.e., $P^T P = QQ^T = I$, an additional regularizer term is included in the loss:

$$R(P, Q) = \left\| P^T P - I \right\|_F^2 + \left\| QQ^T - I \right\|_F^2. \quad (13)$$

This adaptive approach enables the model to dynamically adjust the rank within each LoRA module, effectively managing its parameter counts based on the significance of the weight matrices. However, according to **SoRA** [78], the importance scores used in AdaLoRA is heuristically constructed, which lacks rigorous theoretical motivation. Additionally, both moving average operation and calculation of Eq. 13 introduces extra computation cost during training. To address this, SoRA eliminates the orthogonality premise of P and Q. Instead, a gating unit $g \in \mathbb{R}^r$ between W_{up} and W_{down} is directly applied and optimized:

$$h_{out} = W_{\text{up}}(g \odot (W_{\text{down}} h_{in})), \quad (14)$$

where $\odot$ is Hadamard product. The gate g is updated using a variation of proximal gradient iteration for l_1 loss [103], [104], which has a clear mathematical meaning and do not need the heuristic premise. After training, the zeroed-out gate units are pruned by removing the corresponding columns and rows in W_{down} and W_{up}.

Several subsequent studies have aimed to improve LoRA's performance in various aspects. For instance, **Laplace-LoRA** [81] notices that fine-tuned LLMs often exhibit overconfidence. To enhance the calibration of fine-tuned LLMs, Laplace-LoRA utilizes a Bayesian approach, specifically a post-hoc Laplace approximation [105], [106], to the posterior over the LoRA parameters. **LoRA Dropout** [82] introduces random noises to the learnable low-rank matrices and increases parameter sparsity to reduce the risk of overfitting. **LoRA+** [84] proposes to set different learning rates for the LoRA matrices W_{down} and W_{up}, such that $\eta_{\text{up}} = \lambda \eta_{\text{down}}$ with $\lambda > 1$ fixed and tune η_{down}.

Thanks to the modular design of LoRA, many studies incorporate multiple LoRA modules in their frameworks to enhance performance. For example, **LoRAHub** aggregates various LoRA modules trained on different tasks. Given a handful of examples from a new task, LoRAHub can autonomously compose compatible LoRA modules without human intervention via a gradient-free method Shiwa [107]. **MOELoRA** employs a Mixture-of-Experts (MOE) approach to train LoRA in a multi-task setting, resulting in multiple expert LoRA modules. To retrieve parameters for certain task, MOELoRA utilizes a task-motivated gate function that assigns contribution weights to each expert based on the task ID, and the final parameters is calculated through a weighted sum of all experts.

In addition to LoRA, several other reparameterization techniques are emerging with significant potential. For instance, **Compacter** [71] introduces a light-weight adapter modules by parameterizing the W_{down} and W_{up} as $W = \sum_{i=1}^n A_i \otimes B_i$, where $A_i \in \mathbb{R}^{n \times n}$, $B_i \in \mathbb{R}^{\frac{r}{n} \times \frac{d}{n}}$, and $\otimes$ denotes the Kronecker product. They further decrease the parameter count by designating A_i as shared parameters and reparameterizing B_i using the product of two low-rank matrices, effectively reducing the parameter complexity from $\mathcal{O}(rd)$ to $\mathcal{O}(r+d)$. Related studies, such as **KronA** [72] and **KAdaptation** [73], also employ the Kronecker product to reparameterize adapter weights, aiming to achieve parameter reduction. **HiWi** [59] proposes an adapter fine-tuning method that applies an adapter directly to pre-trained parameters instead of hidden representations as:

$$W' = W + \sigma(WW_{\text{down}})W_{\text{up}}, \quad (15)$$

where W denotes the weights or biases within the Transformer block's feed-forward layer. Notably, during inference, this method computes W' in advance, ensuring that the model's inference latency remains on par with that of traditional full fine-tuning. **VeRA (Vector-based Random Matrix Adaptation)** [74] employs a single pair of frozen low-rank matrices W_{up} and W_{down} that are shared across all layers, and adapts these matrices by learning small, trainable scaling vectors represented as b and d (formally denoted by diagonal matrices Λ_b and Λ_d). Specifically, the reparameterization is given by:

$$h_{out} = W_0 h_{in} + \Lambda_b W_{\text{up}} \Lambda_d W_{\text{down}} h_{in}, \quad (16)$$

where both W_{up} and W_{down} are initialized using a random Gaussian distribution. Similar to LoRA, the scaling vector b is initialized to zeros to ensure that the weight matrix is unaffected during the first forward pass. This method significantly reduces the number of trainable parameters compared to LoRA yet maintains the same performance, enabling the fine-tuning of larger models on a single GPU. **DoRA (Weight-Decomposed Low-Rank Adaptation)** [75] presents a novel approach as illustrated in Figure 8 (c) by decomposing model weights $W_0 \in \mathbb{R}^{d \times k}$ into magnitude and direction as follows:

$$W_0 = m \frac{V}{\|V\|_c} = \|W_0\|_c \frac{W_0}{\|W_0\|_c}, \quad (17)$$

where $m \in \mathbb{R}^{1 \times k}$ is the magnitude vector, $V \in \mathbb{R}^{d \times k}$ is the directional matrix, with $\|\cdot\|_c$ being the vector-wise norm of a matrix across each column. Subsequently, DoRA adopts a unique fine-tuning strategy for m and V. While both are tunable, only V undergoes LoRA reparameterization, defined as:

$$W' = \underline{m} \frac{V + \underline{\Delta V}}{\|V + \underline{\Delta V}\|_c} = \underline{m} \frac{W_0 + \underline{W_{\text{up}} W_{\text{down}}}}{\|W_0 + \underline{W_{\text{up}} W_{\text{down}}}\|_c}, \quad (18)$$

where ΔV is the incremental directional update learned by LoRA, and the underlined parameters denote the trainable parameters. Through this methodology, DoRA consistently outperforms LoRA across various tasks and models, demonstrating its superiority.

D. Hybrid PEFT

The efficacy of various PEFT methods can significantly differ across different tasks. As a result, numerous studies aim to either combine the advantages of diverse PEFT approaches or seek to establish a unified perspective by analyzing the similarities among these methods. For instance, **UniPELT**[90] integrates LoRA, prefix-tuning, and adapters into each Transformer block. To control which PEFT submodules should be activated, they also introduce a gating mechanism. This

mechanism consists of three small FFNs that each produce a scalar value $\mathcal{G} \in (0, 1)$, which is then applied to the LoRA, prefix, and adapter matrices, respectively. Across various setups, UniPELT has consistently shown improvements in accuracy ranging from 1% to 4%. **S4** [91] explores design spaces for several PEFT methods (i.e., Adapter (A), Prefix (P), BitFit (B), and LoRA (L)) to uncover underlying design patterns. After a series experiments, their findings include: (1) Applying the spindle grouping partitioning for Transformer layers, which results in four layer groups G_i for $i \in \{1 \ldots 4\}$. Layers in one group have similar behaviors together, which means should be apply similar PEFT strategies. (2) Allocating the number of trainable parameters to layers uniformly. (3) Tuning all the groups. (4) Assigning different PEFT strategies in different group. The resulting design space that has the best performance is:

$$G_1 : (A, L), G_2 : (A, P), G_3 : (A, P, B), G_4 : (P, B, L)$$

MAM Adapter[26] explores the intrinsic similarity between three additive PEFT methods: adapters, prefix-tuning, and LoRA, which leads to the development of three variants: *Parallel Adapter*, which places adapter layers alongside specific layers (SA or FFN) instead of after them; *Multi-head Parallel Adapter*, which divides the parallel adapter into multiple heads, each affecting the head attention output in SA; and *Scaled Parallel Adapter*, which adds a scaling term after the parallel adapter layer, similar to LoRA. Extensive experimentation revealed that the most effective configuration involves using prefix-tuning in the SA layer and the scaled parallel adapter in the FFN layer, which is called MAM Adapter. **LLM-Adapters** [94] builds an easy-to-use framework that incorporates various PEFT techniques into LLMs. Through comprehensive benchmarking across multiple datasets, the study reveals several key insights: (1) The most effective locations for series adapters, parallel adapters, and LoRA are after the MLP layers, alongside the MLP layers, and simultaneously following the Attention layers and MLP layers, respectively. (2) Smaller LLMs utilizing PEFT can achieve competitive or even superior results on certain tasks when compared to their larger counterparts. (3) With appropriate in-distribution fine-tuning data, smaller models are capable of surpassing larger models in task-specific performance.

Several studies leverage neural architecture search (NAS) to find better PEFT combination approaches. For example, **NOAH** [92] discovers that different PEFT configurations are specifically tailored for different tasks. To address this issue, NOAH employs NAS to identify the most effective PEFT configurations for each dataset. Specifically, NOAH's searching space encompasses three PEFT methods: Adapter, LoRA, and Visual Prompt Tuning (VPT). It utilizes AutoFormer [108], a one-shot NAS algorithm, for the efficient discovery of optimal prompt modules. In a related vein, **AUTOPEFT** [93] first establishes a searching space that includes serial adapters, parallel adapters, and prefix tuning. After that, they proposes an effective NAS methods based on a high-dimensional multi-dimensional Bayesian optimisation [109]. Both NOAH and AUTOPEFT demonstrate the capability of NAS in enhancing PEFT configurations across a variety of tasks.

IV. EFFICIENT PEFT DESIGN

Processing latency and peak memory overhead are pivotal factors to consider from a computational standpoint. This section introduces a key characteristic in LLMs aimed at balancing between latency and memory usage (Section IV-A). Following this, we explore strategies for developing efficient PEFT methods to address computational challenges, including **PEFT pruning** (Section IV-B), **PEFT quantization** (Section IV-C), and **memory-efficient PEFT techniques** (Section IV-D), each designed to enhance model performance while minimizing resource consumption. It is noteworthy that quantization inherently addresses memory overhead concerns. However, given its distinct characteristics, we address these quantization methods separately rather than incorporating them under the memory-efficient PEFT section.

A. KV-cache Management for PEFT Efficiency

The core of the LLMs model lies an autoregressive Transformer model, depicted in Figure 2. When we look at autoregression characteristic, it becomes a major challenge in designing an inference system, because every time a new token is generated, the entire LLM model has to transfer all the weights from different memories to the memory of the graphics processor, which is very unfriendly to single-user task scheduling or multi-user work-load balance. The challenging part of serving the auto-regressive paradigm is all previous sequences have to be cached and saved for the next proceeding iteration, the cached activation generated from the previous sequences is stored as the Key-Value Cache (KV-cache).

The storage of KV-cache will cost both memory space and IO performance, yielding in workload memory-bounded and under-utilizing the computation power of the system. Previous works proposed a series of solutions like KV-cache control management [133] or KV-cache compression [134] to improve throughput or reduce latency. When designing PEFT methods, it is crucial to consider the characteristics of the KV-cache to complement its features. For instance, when applying soft prompts in the inference phase, efficiently leveraging the KV-cache for these additional inputs can help accelerate response times by ensuring prompt-related data is readily accessible.

B. Pruning Strategies for PEFT

The inclusion of pruning can substantially enhance the efficiency of PEFT methods. In particular, **AdapterDrop** [110] explores the removal of adapters from lower transformer layers and multi-task adapters in AdapterFusion [29], which shows that the pruning can improve the training and inference efficiency with minimal decrease in performance. **SparseAdapter** [111] investigates different pruning methods and finds that high sparsity ratios (80%) can outperform standard adapters. Additionally, the *Large-Sparse* configuration, which increases the bottleneck dimension while maintaining a constant parameter budget (e.g., doubling dimensions with a 50% sparsity), substantially enhances the model's capacity, resulting in improved performance. **SPLoRA** [112] adopts channel-based pruning to the LoRA weights W_{down} and W_{up}.
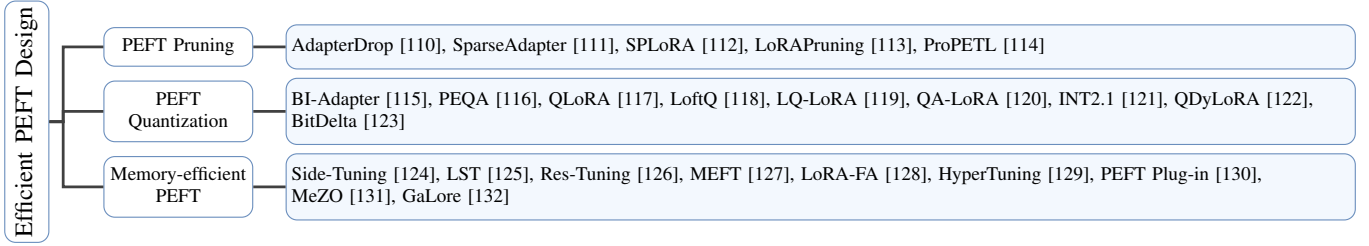
Fig. 9: Taxonomy of Efficient PEFT Design.

This pruning affects not only the source weights W_0, but also the LoRA parameters W_{up} and W_{down}. Similarly, **Lo-RAPruning** [113] adopts structured pruning not only to the pretrained model weights but also to the LoRA weights. In contrast to unstructured LoRA pruning methods, which primarily focus on sparsifying model weights while leaving LoRA weights dense, thus making weight merging challenging to achieve, LoRAPruning enables the weights to be merged easily. Additionally, this work also introduces a novel criterion that utilizes LoRA's gradients as an approximation of the gradients for the pre-trained weights, enabling the estimation of weight importance. **ProPETL** [114] constructs a single shared *prototype* (e.g., adapter, prefix, or LoRA) across layers and tasks. In addition, ProPETL learns binary masks to prune different sub-networks in different layers and tasks. As a result, the parameters can be reused in across layers and tasks, largely increasing the parameter efficiency.

C. Quantization Strategies for PEFT

Quantization serves as another popular technique for improving computational efficiency and reduce memory usage. For example, by investigating the loss landscape of adapters, **BI-Adapter** [115] finds that adapters are resistant to noise in parameter space. Building on this insight, the authors introduce a clustering-based quantization approach. Remarkably, they demonstrate that a 1-bit quantization of adapters not only minimizes storage requirements but also achieve superior performance among all precision settings. **PEQA** (Parameter-Efficient and Quantization-aware Adaptation) [116] uses a two-stage pipeline to achieve parameter-efficient and quantization-aware fine-tuning. In the first stage, the pre-trained FFN weight matrix $W \in \mathbb{R}^{n \times m}$ is quantized to $W = s \cdot \overline{W}$, where $s \in \mathbb{R}^{n \times 1}$ represents per-channel scales and $\overline{W}$ denotes the quantized weight. In the second stage, $\overline{W}$ remains fixed, and fine-tuning is only conducted on s. This approach not only ensures memory efficiency but also facilitates parameter efficiency. **QLoRA** [117] proposes several novel techniques, including a 4-bit NormalFloat, a Double Quantization, and a Paged Optimizers, to backpropagate a 4-bit quantized pretrained language model into LoRA. These techniques enable the fine-tuning for a 65B language model on a single 48GB GPU while maintaining similar performance to the full 16-bit fine-tuning. Similar to the original implementation [70], QLoRA attaches the fixed zero initialized LoRA weights to the quantized pre-trained model as the training start point. However, when applying the extreme low-bit (e.g., 2-bit) quantization, the huge quantization error can adversely impact the

initialization of LoRA fine-tuning, i.e., $quantization(W_0) + W_{down}W_{up} \neq W_0$ where $W_{down} = \mathbf{0}$, which will harm the fine-tuning performance as shown in [127]. To solve this, several quantization strategies are proposed to eliminate the quantization error. For example, **LoftQ** (LoRA-Fine-Tuning-aware Quantization) [118] presents an innovative framework that provides a superior initialization point of quantized backbone weights and LoRA weights for subsequent LoRA fine-tuning. This approach addresses the discrepancies caused by quantization through the optimization of a Frobenius norm objective during network initialization, which takes both the LoRA weights and the quantized pre-trained backbone into consideration. LoftQ exhibits superior performance in 2-bit quantization over QLoRA, as well as greater generalization for downstream tasks. **LQ-LoRA** [119] uses an iterative algorithm inspired by robust principal components analysis [135], [136] which decomposes the weight W_0 such that $W_0 \approx Q + L_1 L_2$ to resolve the inaccuracy caused by the quantization error, where Q is the quantized component which remains fixed and $L_1 L_2$ is the trainable low-rank component. Moreover, this approach leverages integer linear programming to determine a mixed quantization strategy, enabling dynamic quantization configurations for each weight matrix while adhering to a predetermined total bit rate limit. **QA-LoRA** [120] address another limitation of QLoRA, which struggles to preserve its quantized property post fine-tuning. In QLoRA, the quantized pre-trained weight (NF4) have to be recovered to FP16 to match the LoRA weight precision (FP16) during weight merging. Instead, QA-LoRA uses INT4 quantization and introduces group-wise operators to enable quantization during inference stage, therefore improve the efficiency and accuracy compared with QLoRA. **BitDelta** [123] introduces a novel 1-bit post-training quantization method that acts on the weight delta between a fine-tuned model and its underlying pre-trained model. Specifically, given the weight matrices W_{fine} and W_{base} from the fine-tuned and base models respectively, the weight delta $\Delta = W_{fine} - W_{base}$ is binarized as $\hat{\Delta} = \alpha \odot \text{Sign}(\Delta)$. Here, α, a high-precision scalar, is initialized based on the mean absolute delta value $\alpha = \frac{1}{nm} \sum_{ij} |W_{ij}|$, with $\text{Sign}(\cdot)$ indicating the sign of Δ. BitDelta further calibrates the scaling factors via distillation on a compact calibration dataset, while the binary matrices remain unchanged. This approach notably streamlines the deployment of multiple fine-tuned models on shared servers by utilizing a singular full-precision base model alongside efficiently batched 1-bit deltas.

D. Memory-efficient PEFT Methods

Fine-tuning the full LLMs necessitates substantial training memory owing to their considerable size. While most PEFT methods primarily target parameter efficiency, they still incur a significant memory overhead during training because gradient computation and backpropagation are still necessary for these methods. For example, prevalent PEFT techniques such as adapters and LoRA can only reduce memory usage to approximately 70% compared to full model fine-tuning according to some literatures [125], [130]. From a computational perspective, memory efficiency also remains a critical factor that cannot be overlooked.

To improve memory efficiency, various techniques have been developed to minimize the need for caching gradients for the entire LLM during fine-tuning, thereby reducing memory usage. For example, both **Side-Tuning** [124] and **LST** (Ladder-Side Tuning) [125] introduces a learnable network branch parallel to the backbone model. By channeling the backpropagation exclusively through this parallel branch, it circumvents the need to store gradient information for the main model's weights, thus markedly reducing memory requirements during training. Similarly, **Res-Tuning** [126] disentangles the PEFT tuners (e.g., prompt tuning, adapter) from the backbone model. On top of the disentanglement, a memory-efficient fine-tuning framework named Res-Tuning-Bypass is proposed, which generates a bypass network in parallel with the backbone model by removing the data flow from the decoupled tuners to the backbone. This eliminates the requirement for gradient caching within the backbone model during backpropagation. **MEFT** [127] (memory-efficient fine-tuning) is an approach inspired by the *reversible model* [137]. During the training of a reversible model, intermediate activations are not required to be cached in the forward pass. During backpropagation, they can be recalculated from the final output. To save the memory during fine-tuning, MEFT investigates how to transform an LLM to its reversible counterparts without additional pre-training. A critical aspect of this transformation is the careful initialization of newly-introduced parameters in the pre-trained models. MEFT demonstrates the importance of the parameter initialization, and suggests that these parameters must be initialized in a manner that preserves the pre-trained model's starting point, ensuring that the fine-tuning of the modified model achieves performance on par with full fine-tuning methods. With this key consideration, MEFT introduces three distinct methods, each significantly curtailing the memory demands traditionally required for storing activations. **LoRA-FA** [128] addresses a limitation about memory overhead in LoRA fine-tuning. During training, LoRA modules still require high activation memory consumption. This is because, during backpropagation, large input activations must be stored during the forward pass to compute gradients. LoRA-FA resolves this issue by freezing both the pre-trained weights W_0 and the projection-down weights W_{down}, and only updating the projection-up weights W_{up}. Consequently, the input activation h_{in} no longer needs to be stored, as the intermediate activation $W_{\text{down}}h_{in}$ is adequate for gradient computation for W_{up}. Given that $r \ll d$, the memory requirement for activations in LoRA-FA can be significantly reduced.

To further reduce memory usage during fine-tuning, some methods attempt to circumvent backpropagation within LLMs to address this issue. **HyperTuning** [129] employs a Hyper-Model to generate PEFT parameters using only fewshot examples. This approach demonstrates results comparable to those obtained through full model fine-tuning. **PEFT Plug-in** [130] first trains PEFT modules on small language models, which is more memory efficient compared to training on large ones. Subsequently, the research introduces a suite of techniques for seamlessly integrating these trained PEFT modules into LLMs during inference. This strategy effectively circumvents the necessity of gradient-based optimization directly on the larger models, resulting in substantial memory savings. However, it is important to note that both HyperModel and PEFT Plug-in still require additional model training, and this training cost cannot be entirely overlooked. **MeZO** [131] introduces a memory-efficient zeroth-order (ZO) optimizer for LLMs. Unlike conventional PEFT techniques, which rely on backpropagation to compute gradients for updating model parameters, MeZO fine-tunes LLMs through only forward passes. It accomplishes this by employing a ZO gradient estimator to calculate the gradient. Notably, MeZO implements an in-place solution for the classic ZO gradient estimator, effectively mitigating memory consumption during inference execution. This innovative approach allows for efficient fine-tuning of LLMs containing 30 billion parameters on a single GPU with 80GB of memory, all while maintaining performance that is comparable to fine-tuning using backpropagation. Furthermore, it can substantially decrease storage demands in comparison to the traditional PEFT methods such as LoRA and Adapter.

V. PEFT FOR DNNs OF OTHER APPLICATIONS

In Section III, we outlined four categories of PEFT methods along with their improvements. Nonetheless, our discussion did not fully extend to the utilization or adaptation of PEFT techniques beyond traditional architectures (e.g., LLMs) or standard benchmarks (e.g., the GLUE dataset), where the majority of the discussed PEFT methods are applied. Therefore, in this section, we will highlight and discuss several most representative works that leverages PEFT strategies for various downstream tasks. We do not aim to cover all PEFT application scenarios in this section. Our objective is to showcase the significant influence of PEFT within various research domains, and demonstrate how to optimize and tailor general-purpose PEFT methods to achieve enhanced performance in specific models or tasks.

Typically, fine-tuning happens when adapting a pre-trained backbone model to specialized downstream tasks. To this end, this section organizes the discussion around various model architectures, which include: LLM, Vision Transformer (ViT), Vision-Language Alignment Model (VLA), and Diffusion model. Within each architectural category, the discussion is further classify based on different downstream tasks.

A. PEFT for LLMs – Beyond the Basics

Instead of common tasks in NLP such as NLU and NLG, PEFT techniques boast a wide array of applications across

diverse scenarios. PEFT has been successfully implemented in commonsense question answering [138], [139], multi-level implicit discourse relation recognition [140], out-of-distribution detection [141], privacy protection [142], [143], federated learning [144], and social biases mitigation [145]. In this section, we pay more focus on three representative downstream tasks: visual instruction following, continual learning, context window extension.

1) Visual Instruct Following: Several studies, including VL-BART [146], MiniGPT-4 [147], and LLaVA [148], have successfully extended the capabilities of LLMs, initially designed for pure text, to comprehend and generate responses to visual inputs. These enhanced models, namely visual instruct-following LLMs, can process both images and text to produce textual responses, which can be benchmarked on tasks such as image captioning [149], [150], [151], [152] and visual question answering (VQA) [153], [154], [155]. However, these methods fine-tune the entire LLM to learn the visual representations, which can be inefficient in both time and memory. Therefore, its natural to apply PEFT techniques in the fine-tuning of visual instruct-following LLMs. An earlier work **VL-Adapter** [156] directly applies several PEFT methods (Adapter [25], Hyperformer [34] and Compacter [71]) on VL-BART [146] then benchmarks them on several image-text and video-text tasks. Results show that vanilla adapters are the best among them, which can achieve performance on par with full fine-tuning. However, considering the functionality gap between the encoders and decoders in VL-BART, directly assign identical modular modifications will lead to suboptimal performance. Therefore, **VL-PET** [157] selectively integrates PEFT modules into different components of the encoder and decoder. They also introduces a granularity-controlled mechanism for finer-grained control.

To adapt the recently prevalent LLaMA model, **LLaMA-Adapter** [158] prepends a set of learnable prompts (similar to prefix tuning) to the input tokens in LLaMA's higher transformer layers. To avoid the unstable fine-tuning with large loss values at early training stages, instead of the randomly initialized weights of other PEFT methods, LLaMA-Adapter adopts a zero-initialized attention mechanism, which learns a zero-initialized gating factor to adaptively control the contribution of adaptation prompts to the word tokens. This can maintain the fine-tuning starting point the same as the original model and progressively inject new knowledge into the model, where similar idea can be found in MEFT [127] and LoftQ [118] discussed earlier. To represent visual information, LLaMA-Adapter extract multi-scale global image features using CLIP image encoder than projects them to linguistic embedding space. After that, the feature is element-wisely added onto the adaptation prompts at all inserted transformer layers. LLaMA-Adapter only introduces 1.2M learnable parameters in LLaMA-7B, and costs less than one hour for fine-tuning on 8 A100 GPUs. A following work **LLaMA-Adapter V2** [159] demonstrates that the simple multimodal fusion in LLaMA-Adapter cannot generalize to more challenging open-ended multimodal reasoning tasks, where the visual cues tend to dominate the adaptation prompts than the language instruction data. To address this, LLaMA-Adapter V2 decouples the learn-

ing of instruction-following ability (to generate long language responses) and vision-language alignment to avoid interference between visual and language fine-tuning. Specifically, LLaMA-Adapter V2 sets disjoint parameter groups which are respectively learned from image-text pairs and language instruction data. The visual adaptation prompts are inserted in the early stage of LLM, while the language adaptation prompts keeps at the higher transformer layers similar to LLaMA-Adapter. Additionally, LLaMA-Adapter V2 introduces more learnable parameters and several expert systems (e.g., captioning, detection, and OCR) to enhance multimodal performance. **LayerNorm Tuning** [160] adjust only the weights of the LayerNorm within each attention block. This straightforward technique can achieve comparable or even better performance than the finetuning, while offer about 10× more parameter efficiency than LoRA.

2) Continual Learning (CL): CL aims to learn a sequence of new tasks over time within one single model, which has broad application in scenarios such as dialogue systems [161], information extraction systems [162], and question answering systems [163]. The main challenge in CL is catastrophic forgetting [164]. A popular practice, called architecture-based methods, tackles the CL by maintainging task-specific parameters in the model for each new task. Therefore, it's natural to leverage PEFT methods for CL tasks [165], [166], [167], [168]. For example, **AdapterCL** [165] parameterizes each new task using residual adapters. During testing, since the task-id is not provided, AdapterCL uses an entropy-based classifier to select which adapter to use for accomplishing specific task. **CPT** (Continual Prompt Tuning) [166] trains a soft prompt for each task. Instead of training soft prompts from scratch, CPT proposes a series techniques (continual prompt initialization, query fusion, memory replay, and a memory-guided technique) to achieve knowledge transfer from preceding and subsequent tasks. **O-LoRA** (orthogonal low-rank adaptation) [169] employs a strategy of learning distinct tasks within separate low-rank vector subspaces that are kept orthogonal to each other in order to minimize interference. This approace can effectively reducing catastrophic forgetting during the acquisition of new tasks.

3) Context Window Extension: LLMs are typically trained with a pre-defined context size. For example, LLaMA and LLaMA2 have pre-defined context sizes of 2048 and 4096 tokens, respectively. The positional encoding RoPE has weak extrapolation properties [170], which means the performance drops obviously given an input length exceeds the pre-defined context length. To solve this, a naive solution is to fine-tune a pre-trained LLM to longer context. However, this escalates computational costs quadratically with context size, straining memory and processing resources. To address this, **LongLoRA** [171] proposes to fine-tune a pre-trained LLM using LoRA to enlarge the context size. To reduce the perplexity gap between LoRA tuning and full fine-tuning, LongLoRA also opens embedding and normalization layers for training. In order to further improve training efficiency in long context scenario, LongLoRA further introduces a novel shifted sparse attention (S^2-Attn) as an efficient substitute for standard self-attention during training. A subsequent study

LongQLoRA [172] combines the advantages of LongLoRA with QLoRA and Position Interpolation [10] to save GPU memory. This work successfully extends context length of LLaMA2-13B from 4096 to 8192 on a single V100 with 32GB memory. **LLoCO** [173] introduces a pipeline that learns contexts offline through the combination of context compression and LoRA. The process begins by compressing documents into compact contexts, then fine-tuning LLM using LoRA on the compacted context to improve the LLM's ability to accurately extract and utilize information from these compressed representations. During model serving, a standard RAG retriever selects both the compressed document and the most relevant LoRA module, and apply them to the LLM for inference. This approach effectively extends the context window of a 4k token LLaMA2-7B model to handle up to 128k tokens.

In addition to limited training-stage sequence length, real-world system memory constraints introduce another critical bottleneck to the context window. Specifically, the capacity of the KV-cache is curtailed by available system memory. For example, a 30B parameter LLM operating with an input length of 1024 and a batch size of 128 might necessitate up to 180GB for the KV-cache [174], thereby restricting the feasible size of the context window. In response to this, some strategies have resorted to quantizing the KV cache [134], [175], but quantization will certainly compromises performance. To effectively counteract this issue without significant loss, **GEAR** [176] presents a novel approach by employing a low-rank matrix to capture the majority of coherent bases of quantization error, complemented by a sparse matrix that addresses errors from outlier entries, thus efficiently minimizing approximation errors.

B. PEFT for ViTs

ViT [177] has emerged as a powerful backbone model in the recent computer vision community. In ViT model, images are treated as sequences of fixed-size patches analogous to how LLM uses discrete tokens. These patches undergo linear embedding and then receive positional encodings. Subsequently, they are processed through standard Transformer encoders. The training of ViT can be supervised [177], [178] or self-supervised [179], [180], and ViT can achieve superior performance when training with more data and using larger model size [181]. However, such scaling up inevitably escalates training and storage costs. Therefore, similar to LLMs, PEFT widely implemented in various downstream tasks, such as dense prediction [182], continual learning [183], [184], deep metric learning [185]. Here, we focus on two typical tasks to showcase the involvement of PEFT: image classification and video recoginition.

1) Image Classification: Image classification on targeted visual datasets is a very common demand and has extensive applications, while pre-train then fine-tuning paradigm serves as a widespread strategy. A variety of methods leverage PEFT techniques to achieve efficient model tuning [186], [182], [187], [188]. For instance, **AdaptFormer** [187] inserts adapter modules in parallel to the FFN of the original ViT model for visual recognition tasks. **VPT** (Visual Prompt Tuning) [186] prepends a small amount of task-specific parameters into the input sequence of each Transformer layer. When applying ViT to downstream tasks, only these added parameters and the classification head are set to trainable. **The work by** [189] notices that compared with supervised ViT, VPT often underperforms with self-supervised ViT. Further analysis demonstrates that different pre-trained methods and downstream tasks have varying degrees of dependency on transformer blocks at different locations. To tackle this issue, the research introduces adaptable gates for ViT blocks. These gates dynamically modulate the contribution of prompt tokens to ViT blocks, allowing for a more targeted adaptation of the model to the task at hand.

2) Video Recognition: Several works consider the more challenging adaptation problem that transfer ViT to downstream tasks that has a much larger domain gap. For example, **ST-Adapter** (Spatio-Temporal Adapter) [190] and **AIM** [191] both insert adapters layers into pre-trained ViT blocks. Their primary goal is to model spatial-temporal information, thereby enabling efficient adaptation of ViTs from image models to video tasks. Notably, both methodologies have exhibited performance that surpasses traditional full-model fine-tuning approaches.

C. PEFT for VLAs

Vision-Language alignment models (VLA), such as CLIP [192], ALIGN [193], DeCLIP [194], and FLAVA [195], are designed to learn a good image and text features which can be aligned within a unified representation space. Each VLA typically consists of separate image and text encoders that extract respective features. Contrastive learning is leveraged in these models to effective align the image and text features. Fine-tuning is leveraged to improve the performance of VLA in specific dataset or tasks, but fine-tuning the full model is computationally intensive. For instance, fine-tuning CLIP RN50x64 requires a batch size of 32,768 and 18 days of training on 592 V100 GPUs [192]. Moreover, full fine-tuning on smaller datasets often leads to catastrophic forgetting [164]. In response to these challenges, and drawing inspiration from the success of PEFT techniques in NLP, a range of PEFT strategies have been proposed and implemented in VLA models, such as semantic segmentation [196], [197], [198], point cloud understanding [199], [200], [201], [202], video understanding [203], [204], [205], visual reasoning [206], [207], temporal action detection [208], to name a few. This section will focus on one common task that uses VLAs: open-vocabulary image classification.

1) Open-vocabulary Image Classification: In open-vocabulary image classification, earlier works design class-specific prompts, e.g., *a photo of a [CLASS]*, for each category, and ranks images based on their similarity to these textual descriptions. **CoOp** (Context Optimization) [209] replaces the handcrafted text prompt with learnable vectors, while keep the entire VLA fixes during training. **CoCoOp** (Conditional Context Optimization) [210] builds on this by tackling CoOp's limitations in generalizing to unseen classes. It introduces a lightweight neural network that generates an

input-specific context token, dynamically adapting the prompt based on each image, thereby enhancing generalizability, but at the cost of increased computational demands due to the instance-aware operation. **ProGrad** [211] addresses the over-fitting risk in CoOp in few-shot setting by regularizing the soft prompt updates whose gradient is aligned to the general knowledge only updates the prompt whose gradient is aligned (or non-conflicting) to the general knowledge offered by the original prompt. **MaPLe** [212] notes that existing methods learn prompts either in the language or in the vision branch of CLIP, which is not efficient to leverage the multimodal nature of VLAs. To address this, MaPLe proposes branch-aware hierarchical prompts that simultaneously adapt both language and vision branches, and achieves superior performance. **TPT** (test-time prompt tuning) [213] studies prompt tuning on the fly without additional training samples. Specifically, during inference, TPT first augments the input image into various views, which are then utilized to tune the learnable prompts. The primary training objective is to ensure the VLA can generate consistent responses when faced with these differing views. A following work **DiffTPT** [214] further enhances the data diversity of test samples through diffusion models.

In another direction, several studies explores the usage of adapters in VLA. For example, **CLIP-Adapter** [215] integrates residual-style adapters after CLIP's text and visual encoders. Therefore, unlike CoOp and CoCoOp, CLIP-Adapter avoids the gradients backpropagation through CLIP's encoders, leading to reduced computational requirements in terms of both training memory and time. **Tip-Adapter** [216] adopts the same design with CLIP-Adapter. Different from CLIP-Adapter, the weights of adapter is obtained in a training-free manner from a query-key cache model [217], [218] constructed from fewshot supervisions in a non-parametric manner. As a result, Tip-Adapter exhibits great efficiency compared to CLIP-Adapter's SGD training process.

D. PEFT for Diffusion Models

Diffusion models [219], [220] are a class of generative models that learn to generate data by transforming random noise into a structured output by a progressive denoising process. During training, diffusion models learn to reverse the noise added to training data using a denoising network, while in inference, they start from noise, using denoising network to iteratively create data that mirrors the same distribution as the training examples. Diffusion models has various applications [221], [222], [223], [224], [225], while the most notable is stable diffusion [226], which bridges the gap between text and image with its robust capability to generate coherent and contextually relevant images directly from textual descriptions. Numerous studies leverage PEFT techniques to adapt a pre-trained diffusion model for downstream tasks, including accelerating sampling speed [227], [228], text-to-video adaptation [229], [230], text-to-3D adaptation [231], etc. This section mainly focus on two scenarios: integrating additional input modalities beyond mere text-based conditioning, and customizing content generation based on pre-trained diffusion model.

1) Additional Input Control: To incorporate additional input modalities (e.g., layout, keypoints) while retaining the extensive knowledge in the pre-trained model, **GLIGEN** introduces a novel approach, which maintains the original model's weights intact and integrates new, trainable gated Transformer layers [232] that take in the new grounding input. The resulting model can not only accurately represent the grounding conditions but also produce high-quality images. Remarkably, the model can also generalize well to unseen objects during inference. **ControlNet** [233] fine-tunes a trainable copy of the encoding layers from Stable Diffusion while locks its pre-trained parameter weights. The fixed original model and the trainable copy are bridged through zero convolution layers. These layers, starting with zero-initialized weights, are designed to progressively adapt during training, ensuring that harmful noise does not affect the pre-trained features of Stable Diffusion at the beginning of training. This refined model is capable of conditioning on a variety of inputs such as Canny edges, Hough lines, user scribbles, human key points, segmentation maps, shape normals, depths, etc. **Concept Sliders** [234] introduces a plug-and-play LoRA adaptors to allow precise editing of concepts (e.g., age, smiling) within a diffusion model. **T2I-Adapter** [235] introduces a lightweight adapter model designed to align external control signals with the internal knowledge of text-to-image diffusion models. This adapter enables precise manipulation through structural control (e.g., sketch, depth map, semantic segmentation map, and keypose), color control (e.g., hue and color distribution), and integrating various controls by composing multiple adapters.

2) Customized Generation: The effectiveness of text-to-image diffusion models is limited by the user's ability to articulate the desired target through text descriptions. For instance, it is difficult to describe the precise features of an innovative toy car which is not encountered during large-scale model training. Consequently, the objective of customized generation is to enable the model to grasp new concepts from a minimal set of user-supplied images. **Textual Inversion** [236] addresses this by finding a new pseudo-word S_* (similar to soft prompt discussed in Section III-A2) that represent new, specific concepts in the textual embedding space of pre-trained text-to-image diffusion models. The pseudo-word S_* is optimized via the original optimization goal in diffusion models given a small image set (typically 3-5 images) depicting the concept, and the pre-trained model is leaved untouched. During inference, S_* can be treated like any other word and compose with other textual queries (e.g., "a photo of S_* on the beach"). **Custom Diffusion** [237] tackles a more challenging setting: compositional fine-tuning of multiple concepts. It fine-tunes only the W_k, W_v mapping from text to latent features in attention layers, which yields superior performance in multi-concept learning scenarios. Additionally, during fine-tuning, Custom Diffusion prevents model forgetting by introducing a small set of real images with captions akin to the target, alongside employing augmentation for faster convergence and improved results. **IP-Adapter** [238] identifies limitations in current approaches (e.g., ControlNet and T2I-Adapter) which project condition signals into the cross-attention modules. When handling image conditions aiming at controlling con-

tent, these methods unable to generate images faithful to the prompted image. The issue stems from that merging image features and text features within cross-attention layers loses image-specific information, leading to only coarse-grained controllable generation such as image style rather than image content. To overcome this, IP-Adapter introduces a novel decoupled cross-attention mechanism to distinguish between text and image features. IP-Adapter adds an additional cross-attention layer exclusively for image features in each cross-attention layer, and only the parameters of the new cross-attention layers are trained.

VI. SYSTEM DESIGN CHALLENGE FOR PEFT

A. System design for PEFT

In this section, we begin by providing a concise overview of cloud-based PEFT systems. Following this, we present the corresponding metrics employed for evaluating the system performance. Additionally, we present three prospective utilization scenarios to illustrate the challenges in system design.

1) Centralized PEFT Query Serving: Cloud providers have recently introduced a range of LLM services aimed at providing user applications through application programming interfaces (APIs) [239], [240]. These APIs facilitate the seamless integration of many ML functionalities into applications. After receiving one query for one specific downstream task through API, the cloud-based server processes the query with one featured LLM model. Under this scenario, the proposed cloud solution for handling multiple PEFT queries involves storing only a single copy of the LLM and multiple PEFT modules. This single copy maintains multiple branches of PEFT modules, each associated with different PEFT queries. The case study of a state-of-the-art system can be found in Section VI-C. Figure 10 (b) illustrates the computation pattern for multi-query PEFT inference, wherein packed PEFT queries are scheduled and executed according to their deadlines and current system conditions.

2) Serving Metrics: To evaluate the system performance of centralized PEFT query serving, we propose a set of evaluation metrics.

- **System throughput**: Considering PEFT queries as inter and intra tasks, we use tokens per second to measure the system throughput.
- **Memory footprint**: Run-time memory consumption during query serving, the memory utilization comes from both model parameters and KV-cache as mentioned in Section IV-A.
- **Accuracy performance**: Real-world queries normally have different context lengths, and performance with variation length serves as a performance benchmark.
- **Quality of services**: Queries are associated with latency requirements and deadline missing rates are considered as another benchmark.

3) Distributed System for PEFT: Nevertheless, in the contemporary LLM model, personalized tasks are not fully supported with pre-trained models, consequently, extra fine-tuning is required to be executed with the methodologies mentioned in the previous sections. However, a big concern is raised when
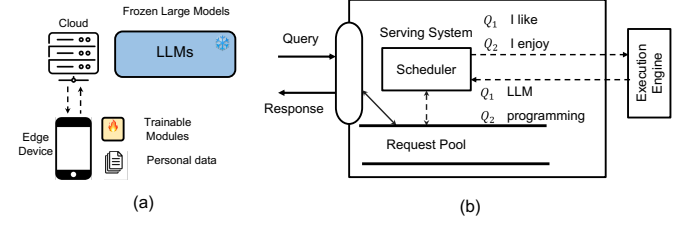

Fig. 10: (a) Distributed-based system computation pattern; (b) centralized PEFT Query inference

we consider giving the datasets to cloud providers since these datasets are personalized.

For this concern, DLoRA [241] presents a distributed PEFT framework. During the PEFT process, the backbone LLM is executed in the cloud servers while the PEFT modules are trained entirely within the user devices. DLoRA scheme is depicted in Figure 10(a).

4) Distributed Metrics: To assess the efficacy of the proposed method, we establish a set of evaluative metrics. For this analysis, and without loss of generality, we adopt language models as the basis for our metric definitions.

- **Accuracy performance**: Performance of the fine-tuned model over the downstream tasks.
- **Compute cost**: The compute cost during forward and backward propagation operations on edge devices.
- **Communication cost**: Refers to the volume of data involved during the transfer of intermediate data between the edge device and the cloud.

5) Multi-PEFT Training: Different from multiple-PEFT serving, tuning with multiple customized PEFTs always involves different backbone LLMs. When contemplating LLM usage across various downstream tasks, pre-trained models typically exhibit subpar performance. A prevalent approach to adapt LLM to diverse tasks involves crafting fine-tuned PEFTs. However, simultaneously tuning multiple PEFTs can pose considerable challenges. Challenges like how to manage memory gradient and model weights storage, and how to design an efficient kernel for batching PEFT training remain unsolved. PEFTs will be categorized based on their PEFT algorithms and backbone LLM models. The design challenge involves how to consolidate multiple PEFTs with the same LLM backbone and multiple different LLM backbones simultaneously.

B. Case study: Offsite-Tuning

We already know that fine-tuning LLM for downstream tasks is challenging for two reasons: dual privacy concerns between cloud server and data owner, and issues with computational resources and efficiency. Firstly, the privacy of both parties is at risk: the weights of large models are often proprietary and not made public. Sharing data with model owners for fine-tuning can lead to data privacy concerns while providing model weights to data proprietors could compromise the ownership of proprietary models. Secondly, even if downstream users have access to pre-trained weights, the stringent hardware requirements make transfer learning impractical for most end users.

To resolve these two issues, Offsite-Tuning [242] proposes a privacy-preserving and efficient transfer learning framework that enables foundational models to adapt to downstream tasks without the need to access the complete model weights. The key insight of Offsite-Tuning is the cloud provider sends an adapter and an emulator to the data proprietor. Then, with the assistance of the emulator, the data proprietor fine-tunes the adapter. The fine-tuned adapter is then sent back to the cloud side, which integrates it into the complete model, creating a fine-tuned foundational model for downstream users.

Offsite-Tuning safeguards the privacy of data proprietors since they do not need to share their training data directly. It also protects the foundational model owners, as the complete model weights are not shared, and the emulator provided is lossy, with significantly degraded performance. Compared to existing fine-tuning methods that require access to the full model weights, Offsite-Tuning is more resource-efficient because it allows for fine-tuning through a compressed emulator without needing the complete model.

C. Case Study: PetS

The PEFT algorithm is notable for its ability to distinguish between modifiable and immutable weights within a model. This characteristic inspires developers to amalgamate diverse LLMs with distinct PEFT techniques into collective units. PetS, as introduced in [243], advocates for a comprehensive approach to managing multiple PEFT tasks by suggesting a unified serving framework. The framework's core advancement lies in the translation of varying PEFT tasks into integrated computation kernels to enhance efficiency. Moreover, PetS pioneers an orchestrated batching approach and a scheduling methodology, aiming to augment system throughput and leverage task parallelism respectively.

As depicted in Figure 11, the PetS framework begins with users registering PEFT tasks through a standardized Application Programming Interface (API). Upon registration, developers are expected to provide the Pre-Trained Model Tag (e.g., LLaMA), PEFT parameters in a compressed format, and the specific PEFT algorithms (e.g., LoRA, Adapter, Bitfit, etc.). These tasks are then endowed with unique identifiers, and the inference engine takes charge of query processing. PetS bifurcates the primary computational workload (e.g., linear layer computations) into three distinct computational operations: (1) Dense Matrix-Vector Multiplication (MVM) leveraging universally accessible, pre-trained weights. (2) Bias vector addition (Vadd), using either common or task-exclusive biases. (3) A combination of Sparse/dense MVM operations employing task-specific PET parameters. A unified pre-trained weight matrix W is employed across PetS, facilitating the batching of initial operations, $X_t \times W$. However, subsequent task-specific computations involving PET parameters, despite being relatively minimal in complexity, are processed individually.

Considering the Adapter and Bitfit tasks as an illustration, both aim at the MLP component of LLMs. The Adapter task integrates additional weight segments, whereas Bitfit adjusts bias elements. The Adapter operation is modeled as $Y = X_{in1} \times (W + W_{ad}) + b_0$, where X_{in1} represents the input for the Adapter task, W and W_{ad} are the original and adapter-specific PEFT weights respectively, and b_0 is the initial bias. The Bitfit operation, on the other hand, is defined as $Y = X_{in2} \times W + b_1$, with b_1 symbolizing the Bitfit-adjustable bias. These operations are further synthesized as $\{Y_1, Y_2\} = \{X_{in1}, X_{in2}\} \times W + \{X_{in1} \times W_{ad}, 0\} + \{b_0, b_1\}$, delineating that the $\{X_{in1}, X_{in2}\} \times W$ part is amenable to batching through MVM, while the $\{b_0, b_1\}$ segment pertains to the Vadd operation.

For tasks like Diff-Pruning III-B, is a little bit different than Bitfit and Adapter. For Diff-Pruning, the computation concerning the shared weight and 'difference' are conducted separately. Then the results are added up, namely

$$X_t \times (W + \delta_t) = X_t \times W + X_t \times \delta_t$$

, here, the W denotes the backbone model weights while δ_t denotes the pruned weights which can be represented as Sparse MVM.

The other challenge PetS proposed is how to schedule different PEFT requests to achieve high performance. PetS scheduler achieves high parallelism through a two-level scheduling policy: Coordinated Batching (CB) and Macro-batch Streaming (MS) as Figure 12 depicts. Through CB, the input queries will first be clustered based on their input length and then grouped based on their shared operator. This is to make sure the same sequence length of queries will be executed without wasting padding. MS strategy will take the grouped queries after coordinated batching and the theoretical latency for different operators as well as the system modeling parameters to generate the best execution order.

D. Parallel PEFT Training Frameworks

a) Design Challenges: Unlike the PetS system, which aims to accommodate flexible multi-PEFT algorithms, S-LoRA [244] and Punica [245] focus solely on facilitating multiple-LoRA blocks for various tasks. Designing multiple PEFT training systems presents key challenges in two main aspects:

- Efficient concurrent execution of multiple PEFT models with the same LLM backbone.
- Designing an efficient system for multi-tenant serving with different LLM backbones.

b) Efficient kernel design: Punica addresses the first challenge by using existing matrix multiplication for the backbone computation and introducing a new CUDA kernel, Segmented Gather Matrix-Vector Multiplication (SGMV), for adding the PEFT add-ons to the backbone computation in a batched manner. This kernel parallelizes the feature-weight multiplication for different requests in the batch and groups requests corresponding to the same PEFT model to increase operational intensity and use GPU Tensor Cores for acceleration.

The second challenge is beyond the computational cost, designing an efficient system architecture that can effectively serve multi-tenant PEFT model workloads on the smallest set of GPUs possible while occupying the least amount of GPU resources is another significant challenge. Punica addresses
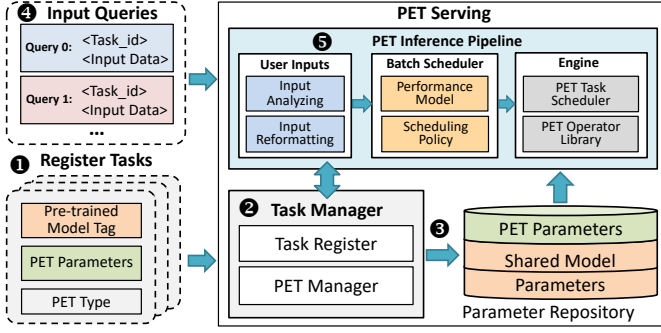
Fig. 11: PetS system overview: (1) Tasks register; (2) Task manager (3) Task schedule; (4) Task serving. (Image is taken from PetS [243])

Fig. 12: Coordinated Batching (CB) Strategy

this by scheduling user requests to active GPUs that already serve or train PEFT models, thereby improving GPU utilization. For older requests, Punica periodically migrates them to consolidate workloads, thus freeing up GPU resources for new requests.

c) Multi-Tenant PEFT design: Designing an efficient system for the multi-tenant PEFT model serving in the Punica framework focuses on addressing several key challenges to maximize hardware utilization and minimize resource consumption. The system aims to consolidate multi-tenant LoRA serving workloads onto the smallest set of GPUs possible. This consolidation is achieved through strategic scheduling of user requests to active GPUs that are already serving or training LoRA models, thereby improving GPU utilization. For older requests, Punica periodically migrates them to consolidate workloads further, thus freeing up GPU resources for new requests. It incorporates on-demand loading of LoRA model weights, which introduces only millisecond-level latency. This feature provides Punica with the flexibility to dynamically consolidate user requests to a small set of GPUs, without being constrained by the specific LoRA models already running on those GPUs. Besides that, Punica identifies that the decode stage is a predominant factor in the cost of model serving, Punica's design primarily focuses on optimizing decode stage performance. Other aspects of model serving leverage straightforward techniques, such as on-demand loading of LoRA model weights, to efficiently manage resource utilization.

VII. CONCLUSION AND FUTURE DIRECTIONS

In the current era dominated by large models and large datasets, PEFT stands out as a highly attractive method for efficiently adapting models to downstream tasks. This technique gains its appeal by addressing the significant challenges posed by traditional full-model fine-tuning, which often places substantial computational and data demands. This survey offers a comprehensive examination of the most recent advancements in PEFT, including algorithmic design, computational efficiency, application scenarios, and system implementation for PEFT. It offers a comprehensive taxonomy and explanation that serves as an excellent guidance and knowledge base, which enables readers of various levels and disciplines to swiftly grasp the core concepts of PEFT.
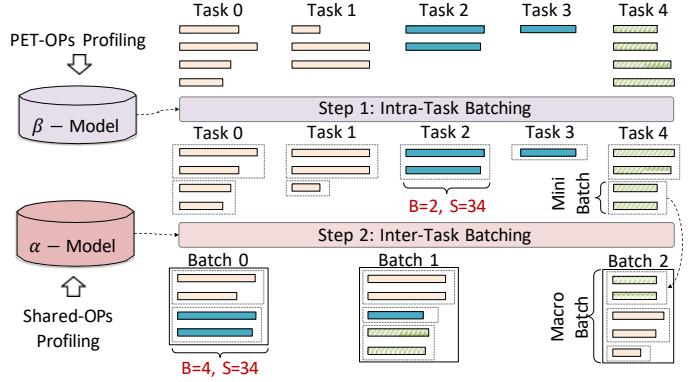
For further research on PEFT, we propose a series of possible directions from both algorithm and system perspectives, hoping to inspire more researchers to engage in further studies in these areas.

A. Simplify hyperparameter tuning

The effectiveness of PEFT is often sensitive to its hyperparameters, such as the bottleneck dimension of the adapter, the rank of LoRA, and the arrangement of various additive PEFT layers. Manually tuning these hyperparameters will cost lots of efforts. Therefore, future efforts could focus on developing methods that are less dependent on manual tuning of these parameters, or automatically find the optimal configuration settings. Several studies [76], [77], [78], [91], [92], [93] have started to address this issue, but there's a need for more simple and efficient solutions optimizing these hyperparameters.

B. Establish a unified benchmark

Despite the existence of libraries like HuggingFace's PEFT [246] and AdapterHub [247], a comprehensive benchmark for PEFT is still lacking. This gap hinders the ability to fairly compare the performance and efficiency of different PEFT approaches. A well-accepted, up-to-date benchmark akin to MMDetection [248] for object detection would enable researchers to validate their methods against a standard set of tasks and metrics, fostering innovation and collaboration within the community.

C. Enhance training efficiency

The presumed parameter efficiency of PEFT does not always consistent with computational and memory savings during training. Given that trainable parameters are intertwined within the pre-trained model's architecture, computing and storing activations and gradients for the full model often become necessary during fine-tuning. This oversight calls for a rethinking of what constitutes efficiency. As outlined in Section IV, potential solutions lie in the integration of model compression techniques such as pruning and quantization, alongside innovations specifically designed to optimize memory during PEFT tuning [249]. Further research into enhancing

the computational efficiency of PEFT methodologies is imperative.

D. Explore scaling laws

The design and effectiveness of PEFT methods originally developed for smaller Transformer models do not necessarily scale with larger models. As the size of foundation models increases, identifying and adapting PEFT strategies that remain effective is crucial. This investigation will aid in customizing PEFT methodologies to suit the evolving landscape of large model architectures.

E. Serve more models and tasks

The rise of large foundation models across various domains presents new opportunities for PEFT. Designing PEFT methods tailored to the unique characteristics of models, such as Sora [250], Mamba [251], and LVM [252], can unlock new application scenarios and opportunities.

F. Enhancing data privacy

Trusting centralized systems to serve or fine-tune personalized PEFT modules is yet another issue for system developers. Multiple types of inversion attacks [253], [254] have been proposed to reconstruct user's data by hijacking the intermediate results. One perspective of future trust-worthy LLM system design involves developing an encryption protocol for both personal data and intermediate training and inference results.

G. PEFT with model compression

Model compression is one of the most effective ways to make LLM executable on resource-limited devices. Yet, the impact of model compression techniques on the performance of PEFT algorithms running on hardware remains another systemic challenge. Common compression techniques such as quantization and pruning necessitate dedicated hardware platforms to expedite the process, and building such hardware platform for compressed models is yet another direction for future research.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang, "Toolqa: A dataset for llm question answering with external tools," *arXiv preprint arXiv:2306.13304*, 2023.

[3] W. Zhu, H. Liu, Q. Dong, J. Xu, L. Kong, J. Chen, L. Li, and S. Huang, "Multilingual machine translation with large language models: Empirical results and analysis," *arXiv preprint arXiv:2304.04675*, 2023.

[4] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. Shaikh, N. Akhtar, J. Wu, and S. Mirjalili, "A survey on large language models: Applications, challenges, limitations, and practical usage," *TechRxiv*, 2023.

[5] B. Xu, X. Liu, H. Shen, Z. Han, Y. Li, M. Yue, Z. Peng, Y. Liu, Z. Yao, and D. Xu, "Gentopia: A collaborative platform for tool-augmented llms," *arXiv preprint arXiv:2308.04030*, 2023.

[6] G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for "mind" exploration of large language model society," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[7] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, 2023.

[8] H. Zhang, X. Liu, and J. Zhang, "Summit: Iterative text summarization via chatgpt," *arXiv preprint arXiv:2305.14835*, 2023.

[9] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[10] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021.

[11] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[12] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? a new dataset for open book question answering," in *EMNLP*, 2018.

[13] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi, "Piqa: Reasoning about physical commonsense in natural language," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[14] M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi, "Socialiqa: Commonsense reasoning about social interactions," *arXiv preprint arXiv:1904.09728*, 2019.

[15] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[16] C. e. a. Clark, "Boolq: Exploring the surprising difficulty of natural yes/no questions," in *NAACL*, 2019.

[17] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.

[18] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv:1803.05457v1*, 2018.

[19] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[20] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag *et al.*, "The" something something" video database for learning and evaluating visual common sense," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5842–5850.

[21] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *2011 International conference on computer vision*. IEEE, 2011, pp. 2556–2563.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[23] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[24] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[25] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[26] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," *arXiv preprint arXiv:2110.04366*, 2021.

[27] Y. Zhu, J. Feng, C. Zhao, M. Wang, and L. Li, "Counter-interference adapter for multilingual machine translation," *arXiv preprint arXiv:2104.08154*, 2021.

[28] T. Lei, J. Bai, S. Brahma, J. Ainslie, K. Lee, Y. Zhou, N. Du, V. Y. Zhao, Y. Wu, B. Li *et al.*, "Conditional adapters: Parameter-efficient transfer learning with fast inference," *arXiv preprint arXiv:2304.04947*, 2023.

[29] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," *arXiv preprint arXiv:2005.00247*, 2020.

[30] Y. Wang, S. Mukherjee, X. Liu, J. Gao, A. H. Awadallah, and J. Gao, "Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models," *arXiv preprint arXiv:2205.12410*, vol. 1, no. 2, p. 4, 2022.

[31] H. Zhao, J. Fu, and Z. He, "Prototype-based hyperadapter for sample-efficient multi-task tuning," *arXiv preprint arXiv:2310.11670*, 2023.

[32] A. Chronopoulou, M. E. Peters, A. Fraser, and J. Dodge, "Adaptersoup: Weight averaging to improve generalization of pretrained language models," *arXiv preprint arXiv:2302.07027*, 2023.

[33] S. He, R.-Z. Fan, L. Ding, L. Shen, T. Zhou, and D. Tao, "Mera: Merging pretrained adapters for few-shot learning," *arXiv preprint arXiv:2308.15982*, 2023.

[34] R. K. Mahabadi, S. Ruder, M. Dehghani, and J. Henderson, "Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks," *arXiv preprint arXiv:2106.04489*, 2021.

[35] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[36] J. Li, W. Aitken, R. Bhambhoria, and X. Zhu, "Prefix propagation: Parameter-efficient tuning for long sequences," *arXiv preprint arXiv:2305.12086*, 2023.

[37] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv preprint arXiv:2110.07602*, 2021.

[38] Z.-R. Zhang, C. Tan, H. Xu, C. Wang, J. Huang, and S. Huang, "Towards adaptive prefix tuning for parameter-efficient language model fine-tuning," *arXiv preprint arXiv:2305.15212*, 2023.

[39] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.

[40] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[41] F. Ma, C. Zhang, L. Ren, J. Wang, Q. Wang, W. Wu, X. Quan, and D. Song, "Xprompt: Exploring the extreme of prompt tuning," *arXiv preprint arXiv:2210.04457*, 2022.

[42] Z. Wu, S. Wang, J. Gu, R. Hou, Y. Dong, V. Vydiswaran, and H. Ma, "Idpg: An instance-dependent prompt generation method," *arXiv preprint arXiv:2204.04497*, 2022.

[43] X. Liu, T. Sun, X. Huang, and X. Qiu, "Late prompt tuning: A late prompt could be better than many prompts," *arXiv preprint arXiv:2210.11292*, 2022.

[44] W. Zhu and M. Tan, "Spt: Learning to selectively insert prompts for better prompt tuning," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 11 862–11 878.

[45] Q. Wang, Y. Mao, J. Wang, H. Yu, S. Nie, S. Wang, F. Feng, L. Huang, X. Quan, Z. Xu *et al.*, "Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 9147–9160.

[46] T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer, "Spot: Better frozen model adaptation through soft prompt transfer," *arXiv preprint arXiv:2110.07904*, 2021.

[47] Y. Su, X. Wang, Y. Qin, C.-M. Chan, Y. Lin, H. Wang, K. Wen, Z. Liu, P. Li, J. Li *et al.*, "On transferability of prompt tuning for natural language processing," *arXiv preprint arXiv:2111.06719*, 2021.

[48] J. Wu, T. Yu, R. Wang, Z. Song, R. Zhang, H. Zhao, C. Lu, S. Li, and R. Henao, "Infoprompt: Information-theoretic soft prompt tuning for natural language understanding," *arXiv preprint arXiv:2306.04933*, 2023.

[49] L. Chen, H. Huang, and M. Cheng, "Ptp: Boosting stability and performance of prompt tuning with perturbation-based regularizer," *arXiv preprint arXiv:2305.02423*, 2023.

[50] Y. Qin, X. Wang, Y. Su, Y. Lin, N. Ding, J. Yi, W. Chen, Z. Liu, J. Li, L. Hou *et al.*, "Exploring universal intrinsic task subspace via prompt tuning," *arXiv preprint arXiv:2110.07867*, 2021.

[51] J.-Y. Choi, J. Kim, J.-H. Park, W.-L. Mok, and S. Lee, "Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 14 306–14 316.

[52] Z. Shi and A. Lipani, "Dept: Decomposed prompt tuning for parameter-efficient fine-tuning," *arXiv preprint arXiv:2309.05173*, 2023.

[53] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1950–1965, 2022.

[54] T. Zadouri, A. Üstün, A. Ahmadian, B. Ermiş, A. Locatelli, and S. Hooker, "Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning," *arXiv preprint arXiv:2309.05444*, 2023.

[55] D. Lian, D. Zhou, J. Feng, and X. Wang, "Scaling & shifting your features: A new baseline for efficient model tuning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 109–123, 2022.

[56] X. Lu, F. Brahman, P. West, J. Jang, K. Chandu, A. Ravichander, L. Qin, P. Ammanabrolu, L. Jiang, S. Ramnath *et al.*, "Inference-time policy adapters (ipa): Tailoring extreme-scale lms without fine-tuning," *arXiv preprint arXiv:2305.15065*, 2023.

[57] D. Guo, A. M. Rush, and Y. Kim, "Parameter-efficient transfer learning with diff pruning," *arXiv preprint arXiv:2012.07463*, 2020.

[58] N. Lawton, A. Kumar, G. Thattai, A. Galstyan, and G. V. Steeg, "Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models," *arXiv preprint arXiv:2305.16597*, 2023.

[59] B. Liao, Y. Meng, and C. Monz, "Parameter-efficient fine-tuning without introducing new latency," *arXiv preprint arXiv:2305.16742*, 2023.

[60] Y.-L. Sung, V. Nair, and C. A. Raffel, "Training neural networks with fixed sparse masks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 193–24 205, 2021.

[61] S. S. S. Das, R. H. Zhang, P. Shi, W. Yin, and R. Zhang, "Unified low-resource sequence labeling by sample-aware dynamic sparse fine-tuning," *arXiv preprint arXiv:2311.03748*, 2023.

[62] A. Ansell, E. M. Ponti, A. Korhonen, and I. Vulić, "Composable sparse fine-tuning for cross-lingual transfer," *arXiv preprint arXiv:2110.07560*, 2021.

[63] Z. Fu, H. Yang, A. M.-C. So, W. Lam, L. Bing, and N. Collier, "On the effectiveness of parameter-efficient fine-tuning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 12 799–12 807.

[64] R. Xu, F. Luo, Z. Zhang, C. Tan, B. Chang, S. Huang, and F. Huang, "Raise a child in large language model: Towards effective and generalizable fine-tuning," *arXiv preprint arXiv:2109.05687*, 2021.

[65] D. Vucetic, M. Tayaranian, M. Ziaeefard, J. J. Clark, B. H. Meyer, and W. J. Gross, "Efficient fine-tuning of bert models on the edge," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 1838–1842.

[66] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.

[67] M. Gheini, X. Ren, and J. May, "Cross-attention is all you need: Adapting pretrained transformers for machine translation," *arXiv preprint arXiv:2104.08771*, 2021.

[68] H. He, J. Cai, J. Zhang, D. Tao, and B. Zhuang, "Sensitivity-aware visual parameter-efficient fine-tuning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 825–11 835.

[69] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020.

[70] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[71] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1022–1035, 2021.

[72] A. Edalati, M. Tahaei, I. Kobyzev, V. P. Nia, J. J. Clark, and M. Rezagholizadeh, "Krona: Parameter efficient tuning with kronecker adapter," *arXiv preprint arXiv:2212.10650*, 2022.

[73] X. He, C. Li, P. Zhang, J. Yang, and X. E. Wang, "Parameter-efficient model adaptation for vision transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, 2023, pp. 817–825.

[74] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano, "Vera: Vector-based random matrix adaptation," *arXiv preprint arXiv:2310.11454*, 2023.

[75] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen, "Dora: Weight-decomposed low-rank adaptation," *arXiv preprint arXiv:2402.09353*, 2024.

[76] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," *arXiv preprint arXiv:2210.07558*, 2022.

[77] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," *arXiv preprint arXiv:2303.10512*, 2023.

[78] N. Ding, X. Lv, Q. Wang, Y. Chen, B. Zhou, Z. Liu, and M. Sun, "Sparse low-rank adaptation of pre-trained language models," *arXiv preprint arXiv:2311.11696*, 2023.

[79] S. Haobo, H. Zhao, S. Majumder, and T. Lin, "Increasing model capacity for free: A simple strategy for parameter efficient fine-tuning," in *The Twelfth International Conference on Learning Representations*, 2023.

[80] R. Zhang, R. Qiang, S. A. Somayajula, and P. Xie, "Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning," *arXiv preprint arXiv:2403.09113*, 2024.

[81] A. X. Yang, M. Robeyns, X. Wang, and L. Aitchison, "Bayesian low-rank adaptation for large language models," *arXiv preprint arXiv:2308.13111*, 2023.

[82] Y. Lin, X. Ma, X. Chu, Y. Jin, Z. Yang, Y. Wang, and H. Mei, "Lora dropout as a sparsity regularizer for overfitting control," *arXiv preprint arXiv:2404.09610*, 2024.

[83] X. Meng, D. Dai, W. Luo, Z. Yang, S. Wu, X. Wang, P. Wang, Q. Dong, L. Chen, and Z. Sui, "Periodiclora: Breaking the low-rank bottleneck in lora optimization," *arXiv preprint arXiv:2402.16141*, 2024.

[84] S. Hayou, N. Ghosh, and B. Yu, "Lora+: Efficient low rank adaptation of large models," *arXiv preprint arXiv:2402.12354*, 2024.

[85] C. Huang, Q. Liu, B. Y. Lin, T. Pang, C. Du, and M. Lin, "Lorahub: Efficient cross-task generalization via dynamic lora composition," *arXiv preprint arXiv:2307.13269*, 2023.

[86] Q. Liu, X. Wu, X. Zhao, Y. Zhu, D. Xu, F. Tian, and Y. Zheng, "Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications," *arXiv preprint arXiv:2310.18339*, 2023.

[87] W. Feng, C. Hao, Y. Zhang, Y. Han, and H. Wang, "Mixture-of-loras: An efficient multitask tuning for large language models," *arXiv preprint arXiv:2403.03432*, 2024.

[88] X. Wu, S. Huang, and F. Wei, "Mixture of lora experts," *arXiv preprint arXiv:2404.13628*, 2024.

[89] D. Li, Y. Ma, N. Wang, Z. Cheng, L. Duan, J. Zuo, C. Yang, and M. Tang, "Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts," *arXiv preprint arXiv:2404.15159*, 2024.

[90] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, W.-t. Yih, and M. Khabsa, "Unipelt: A unified framework for parameter-efficient language model tuning," *arXiv preprint arXiv:2110.07577*, 2021.

[91] J. Chen, A. Zhang, X. Shi, M. Li, A. Smola, and D. Yang, "Parameter-efficient fine-tuning design spaces," *arXiv preprint arXiv:2301.01821*, 2023.

[92] Y. Zhang, K. Zhou, and Z. Liu, "Neural prompt search," 2022.

[93] H. Zhou, X. Wan, I. Vulić, and A. Korhonen, "Autopeft: Automatic configuration search for parameter-efficient fine-tuning," *arXiv preprint arXiv:2301.12132*, 2023.

[94] Z. Hu, Y. Lan, L. Wang, W. Xu, E.-P. Lim, R. K.-W. Lee, L. Bing, and S. Poria, "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models," *arXiv preprint arXiv:2304.01933*, 2023.

[95] S. Hu, Z. Zhang, N. Ding, Y. Wang, Y. Wang, Z. Liu, and M. Sun, "Sparse structure search for parameter-efficient tuning," *arXiv preprint arXiv:2206.07382*, 2022.

[96] A. Petrov, P. H. Torr, and A. Bibi, "When do prompting and prefix-tuning work? a theory of capabilities and limitations," *arXiv preprint arXiv:2310.19698*, 2023.

[97] Y. Wang, J. Chauhan, W. Wang, and C.-J. Hsieh, "Universality and limitations of prompt tuning," *arXiv preprint arXiv:2305.18787*, 2023.

[98] Y. Choi and J.-H. Lee, "Codeprompt: Task-agnostic prefix tuning for program and language generation," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 5282–5297.

[99] H. Wu and X. Shi, "Adversarial soft prompt tuning for cross-domain sentiment analysis," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 2438–2447.

[100] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[101] E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir, "Proving the lottery ticket hypothesis: Pruning is all you need," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6682–6691.

[102] V. Fomenko, H. Yu, J. Lee, S. Hsieh, and W. Chen, "A note on lora," *arXiv preprint arXiv:2404.05086*, 2024.

[103] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[104] A. Chambolle, R. A. De Vore, N.-Y. Lee, and B. J. Lucier, "Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage," *IEEE Transactions on image processing*, vol. 7, no. 3, pp. 319–335, 1998.

[105] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[106] J. Antorán, D. Janz, J. U. Allingham, E. Daxberger, R. R. Barbano, E. Nalisnick, and J. M. Hernández-Lobato, "Adapting the linearised laplace model evidence for modern deep learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 796–821.

[107] J. Liu, A. Moreau, M. Preuss, J. Rapin, B. Roziere, F. Teytaud, and O. Teytaud, "Versatile black-box optimization," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 620–628.

[108] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 270–12 280.

[109] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[110] A. Rücklé, G. Geigle, M. Glockner, T. Beck, J. Pfeiffer, N. Reimers, and I. Gurevych, "Adapterdrop: On the efficiency of adapters in transformers," *arXiv preprint arXiv:2010.11918*, 2020.

[111] S. He, L. Ding, D. Dong, J. Zhang, and D. Tao, "SparseAdapter: An easy approach for improving the parameter-efficiency of adapters," in *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2184–2190. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.160

[112] L. Hedegaard, A. Alok, J. Jose, and A. Iosifidis, "Structured pruning adapters," *arXiv preprint arXiv:2211.10155*, 2022.

[113] M. Zhang, C. Shen, Z. Yang, L. Ou, X. Yu, B. Zhuang *et al.*, "Pruning meets low-rank parameter-efficient fine-tuning," *arXiv preprint arXiv:2305.18403*, 2023.

[114] G. Zeng, P. Zhang, and W. Lu, "One network, many masks: Towards more parameter-efficient transfer learning," *arXiv preprint arXiv:2305.17682*, 2023.

[115] S. Jie, H. Wang, and Z.-H. Deng, "Revisiting the parameter efficiency of adapters from the perspective of precision redundancy," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 217–17 226.

[116] J. Kim, J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee, "Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization," *arXiv preprint arXiv:2305.14152*, 2023.

[117] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[118] Y. Li, Y. Yu, C. Liang, P. He, N. Karampatziakis, W. Chen, and T. Zhao, "Loftq: Lora-fine-tuning-aware quantization for large language models," *arXiv preprint arXiv:2310.08659*, 2023.

[119] H. Guo, P. Greengard, E. P. Xing, and Y. Kim, "Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning," *arXiv preprint arXiv:2311.12023*, 2023.

[120] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. Zhang, and Q. Tian, "Qa-lora: Quantization-aware low-rank adaptation of large language models," *arXiv preprint arXiv:2309.14717*, 2023.

[121] Y. Chai, J. Gkountouras, G. G. Ko, D. Brooks, and G.-Y. Wei, "Int2. 1: Towards fine-tunable quantized large language models with error correction through low-rank adaptation," *arXiv preprint arXiv:2306.08162*, 2023.

[122] H. Rajabzadeh, M. Valipour, T. Zhu, M. Tahaei, H. J. Kwon, A. Ghodsi, B. Chen, and M. Rezagholizadeh, "Qdylora: Quantized dynamic low-rank adaptation for efficient large language model tuning," *arXiv preprint arXiv:2402.10462*, 2024.

[123] J. Liu, G. Xiao, K. Li, J. D. Lee, S. Han, T. Dao, and T. Cai, "Bitdelta: Your fine-tune may only be worth one bit," *arXiv preprint arXiv:2402.10193*, 2024.

[124] J. O. Zhang, A. Sax, A. Zamir, L. Guibas, and J. Malik, "Side-tuning: a baseline for network adaptation via additive side networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 698–714.

[125] Y.-L. Sung, J. Cho, and M. Bansal, "Lst: Ladder side-tuning for parameter and memory efficient transfer learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 991–13 005, 2022.

[126] Z. Jiang, C. Mao, Z. Huang, A. Ma, Y. Lv, Y. Shen, D. Zhao, and J. Zhou, "Res-tuning: A flexible and efficient tuning paradigm via

unbinding tuner from backbone," *arXiv preprint arXiv:2310.19859*, 2023.

[127] B. Liao, S. Tan, and C. Monz, "Make your pre-trained model reversible: From parameter to memory efficient fine-tuning," *arXiv preprint arXiv:2306.00477*, 2023.

[128] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li, "Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning," *arXiv preprint arXiv:2308.03303*, 2023.

[129] J. Phang, Y. Mao, P. He, and W. Chen, "Hypertuning: Toward adapting large language models without back-propagation," in *International Conference on Machine Learning*. PMLR, 2023, pp. 27 854–27 875.

[130] F. Jin, J. Zhang, and C. Zong, "Parameter-efficient tuning for large language model without calculating its gradients," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 321–330.

[131] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora, "Fine-tuning language models with just forward passes," *arXiv preprint arXiv:2305.17333*, 2023.

[132] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian, "Galore: Memory-efficient llm training by gradient low-rank projection," *arXiv preprint arXiv:2403.03507*, 2024.

[133] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.

[134] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré, I. Stoica, and C. Zhang, "Flexgen: High-throughput generative inference of large language models with a single gpu," in *International Conference on Machine Learning*. PMLR, 2023, pp. 31 094–31 116.

[135] T. Zhou and D. Tao, "Godec: Randomized low-rank & sparse matrix decomposition in noisy case," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011.

[136] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," *Advances in neural information processing systems*, vol. 22, 2009.

[137] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," *Advances in neural information processing systems*, vol. 30, 2017.

[138] Y. Huang, Y. Li, Y. Xu, L. Zhang, R. Gan, J. Zhang, and L. Wang, "Mvp-tuning: Multi-view knowledge retrieval with prompt tuning for commonsense reasoning," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 13 417–13 432.

[139] Z. Zhao, L. Hu, H. Zhao, Y. Shao, and Y. Wang, "Knowledgeable parameter efficient tuning network for commonsense question answering," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 9051–9063.

[140] H. Zhao, R. He, M. Xiao, and J. Xu, "Infusing hierarchical guidance into prompt tuning: A parameter-efficient framework for multi-level implicit discourse relation recognition," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 6477–6492.

[141] Y. Ouyang, Y. Cao, Y. Gao, Z. Wu, J. Zhang, and X. Dai, "On prefix-tuning for lightweight out-of-distribution detection," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 1533–1545.

[142] M. S. Ozdayi, C. Peris, J. Fitzgerald, C. Dupuy, J. Majmudar, H. Khan, R. Parikh, and R. Gupta, "Controlling the extraction of memorized data from large language models via prompt-tuning," *arXiv preprint arXiv:2305.11759*, 2023.

[143] G. Xiao, J. Lin, and S. Han, "Offsite-tuning: Transfer learning without full model," *arXiv preprint arXiv:2302.04870*, 2023.

[144] T. Che, J. Liu, Y. Zhou, J. Ren, J. Zhou, V. S. Sheng, H. Dai, and D. Dou, "Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization," *arXiv preprint arXiv:2310.15080*, 2023.

[145] Y. Li, M. Du, X. Wang, and Y. Wang, "Prompt tuning pushes farther, contrastive learning pulls closer: A two-stage approach to mitigate social biases," *arXiv preprint arXiv:2307.01595*, 2023.

[146] J. Cho, J. Lei, H. Tan, and M. Bansal, "Unifying vision-and-language tasks via text generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1931–1942.

[147] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *arXiv preprint arXiv:2304.10592*, 2023.

[148] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *arXiv preprint arXiv:2304.08485*, 2023.

[149] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7008–7024.

[150] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4651–4659.

[151] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 652–663, 2016.

[152] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Computing Surveys (CsUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[153] P. Wang, Q. Wu, C. Shen, A. Dick, and A. Van Den Hengel, "Fvqa: Fact-based visual question answering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 10, pp. 2413–2427, 2017.

[154] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, and A. Van Den Hengel, "Visual question answering: A survey of methods and datasets," *Computer Vision and Image Understanding*, vol. 163, pp. 21–40, 2017.

[155] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2425–2433.

[156] Y.-L. Sung, J. Cho, and M. Bansal, "Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5227–5237.

[157] Z.-Y. Hu, Y. Li, M. R. Lyu, and L. Wang, "Vl-pet: Vision-and-language parameter-efficient tuning via granularity control," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3010–3020.

[158] R. Zhang, J. Han, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, P. Gao, and Y. Qiao, "Llama-adapter: Efficient fine-tuning of language models with zero-init attention," *arXiv preprint arXiv:2303.16199*, 2023.

[159] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue *et al.*, "Llama-adapter v2: Parameter-efficient visual instruction model," *arXiv preprint arXiv:2304.15010*, 2023.

[160] B. Zhao, H. Tu, C. Wei, J. Mei, and C. Xie, "Tuning layernorm in attention: Towards efficient multi-modal llm finetuning," *arXiv preprint arXiv:2312.11420*, 2023.

[161] S. Lee, "Toward continual learning for conversational agents," *arXiv preprint arXiv:1712.09943*, 2017.

[162] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan, "A survey of web information extraction systems," *IEEE transactions on knowledge and data engineering*, vol. 18, no. 10, pp. 1411–1428, 2006.

[163] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin, "End-to-end open-domain question answering with bertserini," *arXiv preprint arXiv:1902.01718*, 2019.

[164] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[165] A. Madotto, Z. Lin, Z. Zhou, S. Moon, P. Crook, B. Liu, Z. Yu, E. Cho, and Z. Wang, "Continual learning in task-oriented dialogue systems," *arXiv preprint arXiv:2012.15504*, 2020.

[166] Q. Zhu, B. Li, F. Mi, X. Zhu, and M. Huang, "Continual prompt tuning for dialog state tracking," *arXiv preprint arXiv:2203.06654*, 2022.

[167] Y. Dai, H. Lang, Y. Zheng, F. Huang, L. Si, and Y. Li, "Lifelong learning for question answering with hierarchical prompts," *arXiv preprint arXiv:2208.14602*, 2022.

[168] Z. Liang, F. Wei, Y. Jie, Y. Qian, Z. Hao, and B. Han, "Prompts can play lottery tickets well: Achieving lifelong information extraction via lottery prompt tuning," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 277–292.

[169] X. Wang, T. Chen, Q. Ge, H. Xia, R. Bao, R. Zheng, Q. Zhang, T. Gui, and X. Huang, "Orthogonal subspace learning for language model continual learning," *arXiv preprint arXiv:2310.14152*, 2023.

[170] S. Chen, S. Wong, L. Chen, and Y. Tian, "Extending context window of large language models via positional interpolation," *arXiv preprint arXiv:2306.15595*, 2023.

[171] Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia, "Longlora: Efficient fine-tuning of long-context large language models," *arXiv preprint arXiv:2309.12307*, 2023.

[172] J. Yang, "Longqlora: Efficient and effective method to extend context length of large language models," *arXiv preprint arXiv:2311.04879*, 2023.

[173] S. Tan, X. Li, S. Patil, Z. Wu, T. Zhang, K. Keutzer, J. E. Gonzalez, and R. A. Popa, "Lloco: Learning long contexts offline," *arXiv preprint arXiv:2404.07979*, 2024.

[174] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett *et al.*, "H2o: Heavy-hitter oracle for efficient generative inference of large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[175] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[176] H. Kang, Q. Zhang, S. Kundu, G. Jeong, Z. Liu, T. Krishna, and T. Zhao, "Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm," *arXiv preprint arXiv:2403.05527*, 2024.

[177] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale. arxiv 2020," *arXiv preprint arXiv:2010.11929*, 2010.

[178] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *arXiv preprint arXiv:2106.10270*, 2021.

[179] X. Chen, S. Xie, and K. He, "An empirical study of training self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9640–9649.

[180] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[181] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin *et al.*, "Scaling vision transformers to 22 billion parameters," in *International Conference on Machine Learning*. PMLR, 2023, pp. 7480–7512.

[182] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision transformer adapter for dense predictions," *arXiv preprint arXiv:2205.08534*, 2022.

[183] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister, "Learning to prompt for continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 139–149.

[184] Q. Gao, C. Zhao, Y. Sun, T. Xi, G. Zhang, B. Ghanem, and J. Zhang, "A unified continual learning framework with general parameter-efficient tuning," *arXiv preprint arXiv:2303.10070*, 2023.

[185] L. Ren, C. Chen, L. Wang, and K. Hua, "Learning semantic proxies from visual prompts for parameter-efficient fine-tuning in deep metric learning," *arXiv preprint arXiv:2402.02340*, 2024.

[186] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European Conference on Computer Vision*. Springer, 2022, pp. 709–727.

[187] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo, "Adaptformer: Adapting vision transformers for scalable visual recognition," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 664–16 678, 2022.

[188] S. Jie and Z.-H. Deng, "Convolutional bypasses are better vision transformer adapters," *arXiv preprint arXiv:2207.07039*, 2022.

[189] S. Yoo, E. Kim, D. Jung, J. Lee, and S. Yoon, "Improving visual prompt tuning for self-supervised vision transformers," *arXiv preprint arXiv:2306.05067*, 2023.

[190] J. Pan, Z. Lin, X. Zhu, J. Shao, and H. Li, "St-adapter: Parameter-efficient image-to-video transfer learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26 462–26 477, 2022.

[191] T. Yang, Y. Zhu, Y. Xie, A. Zhang, C. Chen, and M. Li, "Aim: Adapting image models for efficient video action recognition," *arXiv preprint arXiv:2302.03024*, 2023.

[192] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[193] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International conference on machine learning*. PMLR, 2021, pp. 4904–4916.

[194] Y. Li, F. Liang, L. Zhao, Y. Cui, W. Ouyang, J. Shao, F. Yu, and J. Yan, "Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm," *arXiv preprint arXiv:2110.05208*, 2021.

[195] A. Singh, R. Hu, V. Goswami, G. Couairon, W. Galuba, M. Rohrbach, and D. Kiela, "Flava: A foundational language and vision alignment model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 638–15 650.

[196] M. Xu, Z. Zhang, F. Wei, H. Hu, and X. Bai, "Side adapter network for open-vocabulary semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2945–2954.

[197] Q. Yu, J. He, X. Deng, X. Shen, and L.-C. Chen, "Convolutions die hard: Open-vocabulary segmentation with single frozen convolutional clip," *arXiv preprint arXiv:2308.02487*, 2023.

[198] Z. Xu, Z. Chen, Y. Zhang, Y. Song, X. Wan, and G. Li, "Bridging vision and language encoders: Parameter-efficient tuning for referring image segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 503–17 512.

[199] R. Zhang, Z. Guo, W. Zhang, K. Li, X. Miao, B. Cui, Y. Qiao, P. Gao, and H. Li, "Pointclip: Point cloud understanding by clip," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8552–8562.

[200] X. Zhu, R. Zhang, B. He, Z. Guo, Z. Zeng, Z. Qin, S. Zhang, and P. Gao, "Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2639–2650.

[201] Z. Wang, X. Yu, Y. Rao, J. Zhou, and J. Lu, "P2p: Tuning pre-trained image models for point cloud analysis with point-to-pixel prompting," *Advances in neural information processing systems*, vol. 35, pp. 14 388–14 402, 2022.

[202] T. Huang, B. Dong, Y. Yang, X. Huang, R. W. Lau, W. Ouyang, and W. Zuo, "Clip2point: Transfer clip to point cloud classification with image-depth pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 157–22 167.

[203] C. Ju, T. Han, K. Zheng, Y. Zhang, and W. Xie, "Prompting visual-language models for efficient video understanding," in *European Conference on Computer Vision*. Springer, 2022, pp. 105–124.

[204] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling, "Expanding language-image pretrained models for general video recognition," in *European Conference on Computer Vision*. Springer, 2022, pp. 1–18.

[205] Z. Lin, S. Geng, R. Zhang, P. Gao, G. de Melo, X. Wang, J. Dai, Y. Qiao, and H. Li, "Frozen clip models are efficient video learners," in *European Conference on Computer Vision*. Springer, 2022, pp. 388–404.

[206] Z. Han, F. Zhu, Q. Lao, and H. Jiang, "Zero-shot referring expression comprehension via structural similarity between images and captions," *arXiv preprint arXiv:2311.17048*, 2023.

[207] S. Doveh, A. Arbelle, S. Harary, E. Schwartz, R. Herzig, R. Giryes, R. Feris, R. Panda, S. Ullman, and L. Karlinsky, "Teaching structured vision & language concepts to vision & language models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2657–2668.

[208] S. Nag, X. Zhu, Y.-Z. Song, and T. Xiang, "Zero-shot temporal action detection via vision-language prompting," in *European Conference on Computer Vision*. Springer, 2022, pp. 681–697.

[209] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, 2022.

[210] ——, "Conditional prompt learning for vision-language models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 816–16 825.

[211] B. Zhu, Y. Niu, Y. Han, Y. Wu, and H. Zhang, "Prompt-aligned gradient for prompt tuning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 15 659–15 669.

[212] M. U. Khattak, H. Rasheed, M. Maaz, S. Khan, and F. S. Khan, "Maple: Multi-modal prompt learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 113–19 122.

[213] M. Shu, W. Nie, D.-A. Huang, Z. Yu, T. Goldstein, A. Anandkumar, and C. Xiao, "Test-time prompt tuning for zero-shot generalization in vision-language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 274–14 289, 2022.

[214] C.-M. Feng, K. Yu, Y. Liu, S. Khan, and W. Zuo, "Diverse data augmentation with diffusions for effective test-time prompt tuning," in

Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 2704–2714.

[215] P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao, "Clip-adapter: Better vision-language models with feature adapters," *International Journal of Computer Vision*, pp. 1–15, 2023.

[216] R. Zhang, R. Fang, W. Zhang, P. Gao, K. Li, J. Dai, Y. Qiao, and H. Li, "Tip-adapter: Training-free clip-adapter for better vision-language modeling," *arXiv preprint arXiv:2111.03930*, 2021.

[217] E. Orhan, "A simple cache model for image recognition," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[218] E. Grave, M. M. Cisse, and A. Joulin, "Unbounded cache model for online language modeling with open vocabulary," *Advances in neural information processing systems*, vol. 30, 2017.

[219] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[220] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[221] Z. Han, Y. Wang, L. Zhou, P. Wang, B. Yan, J. Zhou, Y. Wang, and D. Shen, "Contrastive diffusion model with auxiliary guidance for coarse-to-fine pet reconstruction," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2023, pp. 239–249.

[222] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[223] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[224] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[225] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 500–22 510.

[226] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[227] S. Luo, Y. Tan, S. Patil, D. Gu, P. von Platen, A. Passos, L. Huang, J. Li, and H. Zhao, "Lcm-lora: A universal stable-diffusion acceleration module," *arXiv preprint arXiv:2311.05556*, 2023.

[228] W. Chai, D. Zheng, J. Cao, Z. Chen, C. Wang, and C. Ma, "Speedupnet: A plug-and-play hyper-network for accelerating text-to-image diffusion models," *arXiv preprint arXiv:2312.08887*, 2023.

[229] J. Z. Wu, Y. Ge, X. Wang, S. W. Lei, Y. Gu, Y. Shi, W. Hsu, Y. Shan, X. Qie, and M. Z. Shou, "Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7623–7633.

[230] Z. Xing, Q. Dai, H. Hu, Z. Wu, and Y.-G. Jiang, "Simda: Simple diffusion adapter for efficient video generation," *arXiv preprint arXiv:2308.09710*, 2023.

[231] B. Zeng, S. Li, Y. Feng, H. Li, S. Gao, J. Liu, H. Li, X. Tang, J. Liu, and B. Zhang, "Ipdreamer: Appearance-controllable 3d object generation with image prompts," *arXiv preprint arXiv:2310.05375*, 2023.

[232] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 716–23 736, 2022.

[233] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3836–3847.

[234] R. Gandikota, J. Materzynska, T. Zhou, A. Torralba, and D. Bau, "Concept sliders: Lora adaptors for precise control in diffusion models," *arXiv preprint arXiv:2311.12092*, 2023.

[235] C. Mou, X. Wang, L. Xie, Y. Wu, J. Zhang, Z. Qi, Y. Shan, and X. Qie, "T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models," *arXiv preprint arXiv:2302.08453*, 2023.

[236] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personal-izing text-to-image generation using textual inversion," *arXiv preprint arXiv:2208.01618*, 2022.

[237] N. Kumari, B. Zhang, R. Zhang, E. Shechtman, and J.-Y. Zhu, "Multi-concept customization of text-to-image diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1931–1941.

[238] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang, "Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models," *arXiv preprint arXiv:2308.06721*, 2023.

[239] OpenAI, "Gpt-4," in *https://openai.com/gpt-4*, 2023.

[240] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[241] C. Gao and S. Q. Zhang, "Dlora: Distributed parameter-efficient fine-tuning solution for large language model," *arXiv preprint arXiv:2404.05182*, 2024.

[242] G. Xiao, J. Lin, and S. Han, "Offsite-tuning: Transfer learning without full model," *arXiv preprint arXiv:2302.04870*, 2023.

[243] Z. Zhou, X. Wei, J. Zhang, and G. Sun, "{PetS}: A unified framework for {Parameter-Efficient} transformers serving," in *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 2022, pp. 489–504.

[244] Y. Sheng, S. Cao, D. Li, C. Hooper, N. Lee, S. Yang, C. Chou, B. Zhu, L. Zheng, K. Keutzer *et al.*, "S-lora: Serving thousands of concurrent lora adapters," *arXiv preprint arXiv:2311.03285*, 2023.

[245] L. Chen, Z. Ye, Y. Wu, D. Zhuo, L. Ceze, and A. Krishnamurthy, "Punica: Multi-tenant lora serving," *arXiv preprint arXiv:2310.18547*, 2023.

[246] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[247] C. Poth, H. Sterz, I. Paul, S. Purkayastha, L. Engländer, T. Imhof, I. Vulić, S. Ruder, I. Gurevych, and J. Pfeiffer, "Adapters: A unified library for parameter-efficient and modular transfer learning," 2023.

[248] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[249] S. Q. Zhang, T. Tambe, N. Cuevas, G.-Y. Wei, and D. Brooks, "Camel: Co-designing ai models and embedded drams for efficient on-device learning," *arXiv preprint arXiv:2305.03148*, 2023.

[250] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh, "Video generation models as world simulators," 2024. [Online]. Available: https://openai.com/research/video-generation-models-as-world-simulators

[251] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[252] Y. Bai, X. Geng, K. Mangalam, A. Bar, A. Yuille, T. Darrell, J. Malik, and A. A. Efros, "Sequential modeling enables scalable learning for large vision models," *arXiv preprint arXiv:2312.00785*, 2023.

[253] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4829–4837.

[254] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 148–162.