A Survey on Memory-Efficient Fine-Tuning for Large Language Models

Yeachan Kim[1]* **Mingyu Lee**[2]* **SangKeun Lee**[2,3]
[1]Division of Language & AI, Hankuk University of Foreign Studies, Seoul, Republic of Korea
[2]Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea
[3]Department of Computer Science and Engineering, Korea University, Seoul, Republic of Korea
yeachan@hufs.ac.kr, decon9201@korea.ac.kr, yalphy@korea.ac.kr

Abstract

Fine-tuning large language models (LLMs) is a crucial process to align them with human intentions, yet this process remains memory-intensive, varying across tasks and model architectures. These huge and variable memory costs complicate scaling and deployment of LLMs, especially on limited hardware. However, existing surveys on memory efficiency are often either superficial or too narrow in scope, typically focusing on specific subfields. To address this gap, this survey presents the first systematic review of memory-efficient fine-tuning (MEFT) tailored for LLMs. To structure the research landscape, we first categorize existing approaches by their optimization environments (i.e., model itself and systems) and further classify model-based approaches by their specific optimization targets. We also discuss evaluation strategies for assessing MEFT methods and provide empirical analyses. By highlighting challenges and future directions based on current methods, this survey aims to serve as a practical guide for developing MEFT methods.

1 Introduction

Large language models (LLMs) have revolutionized a wide range of research fields far beyond language processing. While these LLMs demonstrate remarkable capabilities, they still exhibit misaligned behaviors with human intentions and task requirements. Fine-tuning has therefore emerged as a crucial approach to steer LLMs toward desired behaviors. However, fine-tuning LLMs requires considerable memory costs. For instance, fine-tuning the LLaMA-70B on multiple sentences exceeds 700GB of memory (Zhang et al., 2024b), which poses significant constraints on commodity hardware. Furthermore, the memory requirements are not static but highly dynamic depending on the task setups (e.g., sequence lengths, batch size).

Despite this complexity and the growing interest in memory-efficient fine-tuning (MEFT), a comprehensive survey remains absent. Existing surveys either focus narrowly on a single subfield, e.g., model compression (Zhu et al., 2024b) or parameter-efficient fine-tuning (Ding et al., 2022; Lialin et al., 2023), or they address MEFT superficially within the broader context of efficiency (Treviso et al., 2023; Wan et al., 2024). Consequently, practitioners and researchers may lack a unified view of MEFT—including memory component, accuracy, and computational cost-across variable configurations. This gap underscores the pressing need for a dedicated survey on MEFT.

This survey fills this gap by providing a comprehensive overview of MEFT, with a particular focus on LLMs. We structure the research landscape by first categorizing existing methods by their optimization environments (the model itself and the system) and then further classifying them by their primary optimization targets (activations, optimizer states, and model weights). Moreover, we discuss performance metrics and benchmarks used to evaluate these MEFT approaches[1]. We then present an empirical survey to assess the strengths and weaknesses of the current methods and share potential future research directions in this rapidly evolving field.

2 Taxonomy of MEFT Approaches

Memory efficiency can be achieved through two primary strategies: (i) reducing the model's intrinsic memory usages, or (ii) distributing the memory load across multiple devices. Accordingly, we first categorize MEFT methodologies into **model-level** and **system-level** optimizations. Figure 1 provides a structured overview starting from this categorization.

Model-level Optimization Model-level methods are further categorized into three groups based on the training memory components they target:

- **Activations:** Activations stored for backpropagation create a major memory bottleneck. Methods in this category reduce this cost by recomputing

* These authors contributed equally to this work.

[1]This survey does not explicitly denote specific methods for pre-training or post-training, as fine-tuning methods can be naturally applied to different phases.

Transactions of the Association for Computational Linguistics, vol. 14, pp. 960–982, 2026. https://doi.org/10.1162/TACL.a.692
Action Editor: Tao Ge. Submission batch: 5/2025; Revision batch: 8/2025; Published 6/2026.

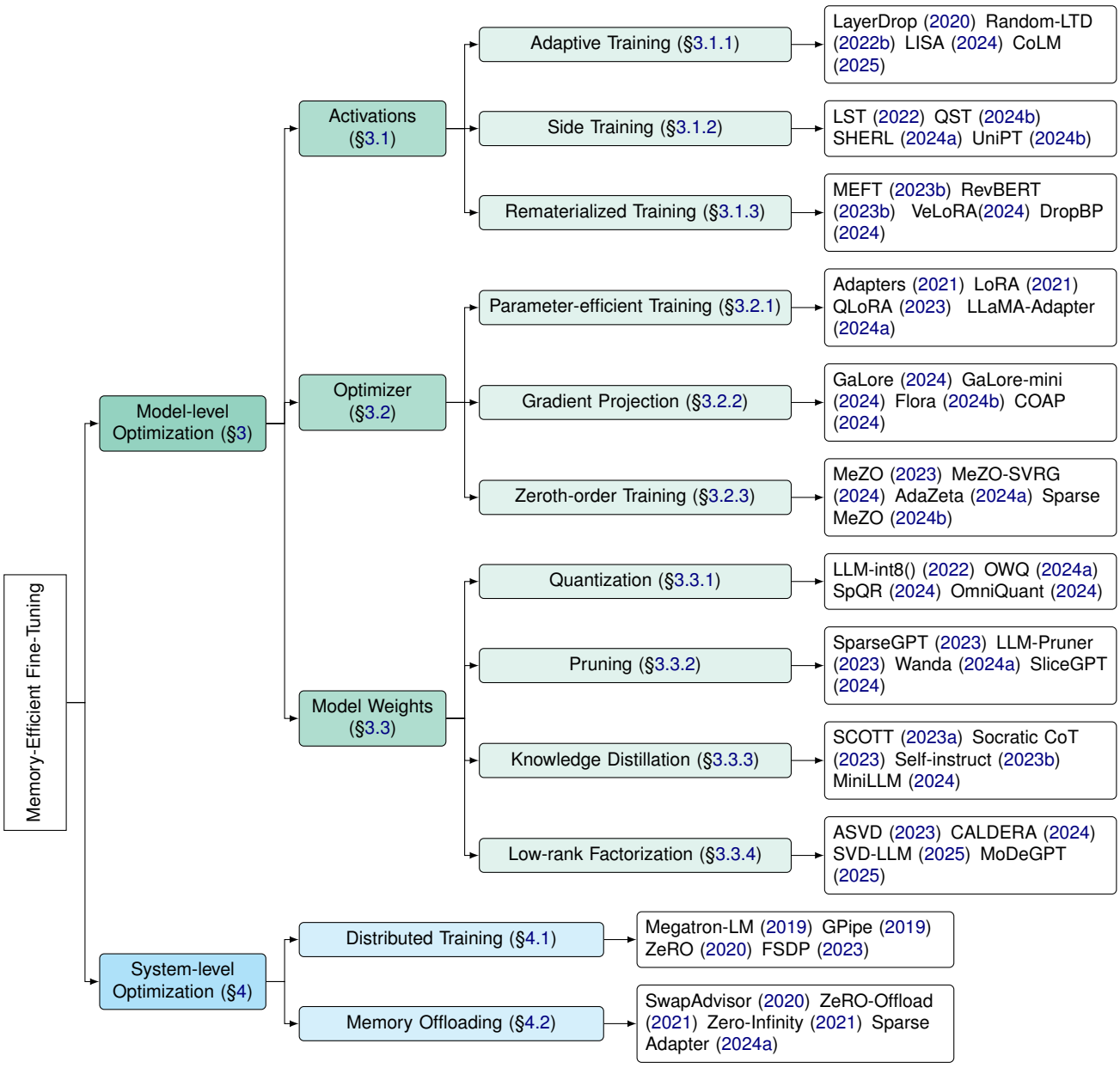


Figure 1: **Research taxonomy.** Taxonomy of MEFT methods for LLMs.

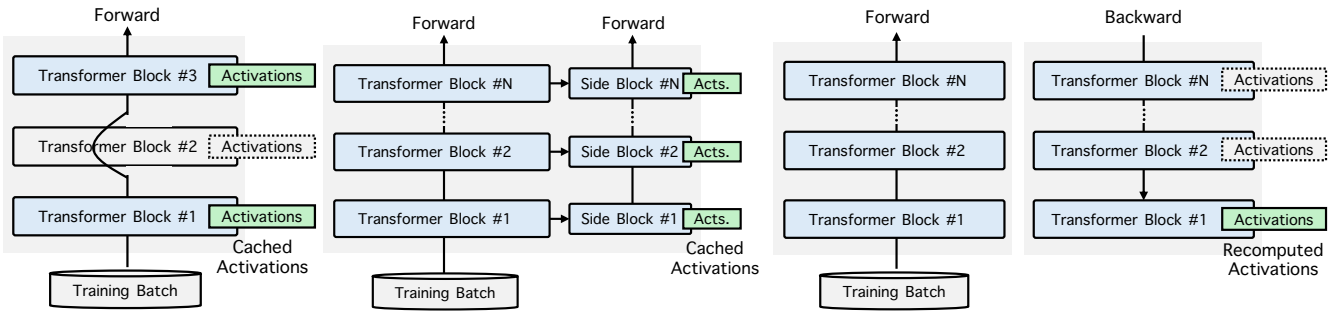
them when needed (Rematerialization), optimizing lightweight auxiliary networks instead of LLMs (Side training), or adaptively optimizing computations (Adaptive training).

- **Optimizer:** Optimizers like Adam require memory to store states (e.g., momentum) for each parameter. Methods in this category reduce this overhead by updating only a subset of parameters (Parameter-efficient training), projecting gradients (Gradient projection), or avoiding backpropagation altogether (Zeroth-order optimization).
- **Model Weights:** The billions of high-precision parameters are a primary memory burden. Methods in this category directly shrink the model size by reducing the numerical precision of weights (Quantization) or decreasing the parameter count (Pruning, Knowledge distillation, Low-rank factorization).

Note that these categories are not mutually exclusive and can often be combined for greater efficiency.

System-level Optimization These methods are further categorized based on their strategy for managing memory burden beyond a single device:

- **Distributed training:** This strategy partitions the model, data, or training process across multiple computing devices (e.g., GPUs, TPUs) to execute them in parallel.
- **Memory offloading:** This technique dynamically moves transient components—such as optimizer states and activations—from the limited GPU memory to a larger but slower storage like CPU memory or NVMe SSDs.



(a) Adaptive Training (b) Side Training (c) Rematerialized Training

Figure 2: **Illustrations of research for Activations.** (a) LayerDrop (Fan et al., 2020) for adaptive training; (b) LST (Sung et al., 2022) for side training; (c) RevBERT (Liao et al., 2023b) for rematerialized training.

3 Model-based Optimization

3.1 Activations

Training phase requires retaining intermediate activations for gradient computation during backpropagation. These memory usages can be reduced by adaptively decreasing computational operations, training auxiliary networks while freezing LLMs, or recomputing or bypassing activations.

3.1.1 Adaptive Training

The number of retained activations is closely associated with the number of computational operations in LLMs. Accordingly, adaptively reducing these operations can reduce the number of retained activations. In the current literature, adaptivity can emerge in three aspects: **data** (i.e., batch computation), **tokens** (i.e., sequence computation), and **layers** (i.e., layer-wise computation).

Data-wise adaptivity For data-level adaptivity, CoLM (Nguyen et al., 2025) achieves memory efficiency by selectively processing data through *coreset* selection (Yang et al., 2023), ensuring that training with an adaptively reduced batch yields gradients similar to those from the original batch. Likewise, online selection approaches based on perplexity (Marion et al., 2023) and submodular facility location (Bhatt et al., 2024) also enhance memory efficiency by compressing each batch into only a few representative samples.

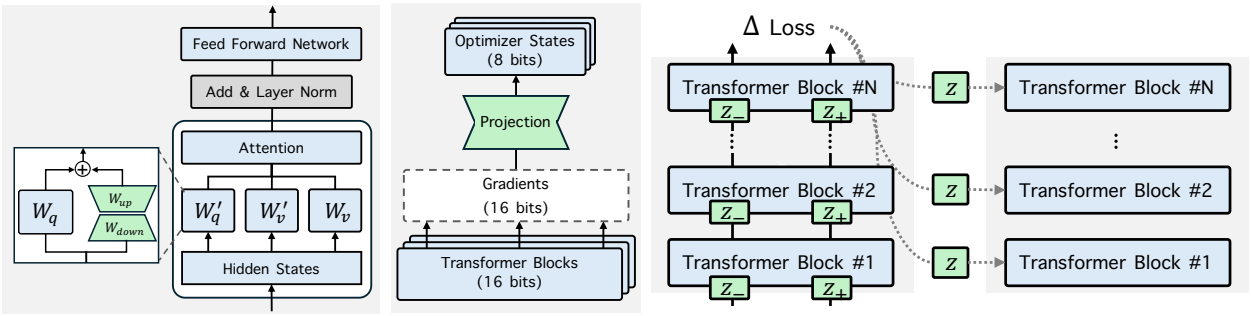
Token-wise adaptivity The number of tokens processed in layers is strongly correlated with the number of computational operations. Consequently, reducing the tokens used in each transformer block provides memory efficiency in terms of activations. For instance, PoWER-BERT (Goyal et al., 2020) and LazyLLM (Fu et al., 2024b) introduces a token pruning based on attention maps. Random-LTD (Yao et al., 2022b) randomly samples tokens in each layer without a specific selection mechanism. However, not all token pruning methods guarantee memory savings; in some cases, learnable approaches (Guan et al., 2022; Kim et al., 2023b) may actually increase memory usage, since these methods were originally designed to accelerate inference rather than MEFT.

Layer-wise adaptivity Similar to token-level adaptation methods, skipping computations at the layer level can also reduce the number of retained activations. For instance, LayerDrop (Fan et al., 2020) randomly drops entire transformer blocks for more efficient inference. Likewise, LISA (Pan et al., 2024) randomly freezes certain layers, guided by each layer's convergence. Similarly, IST (Yao et al., 2024) selectively trains a small subset of important layers based on their contributions to the training losses.

3.1.2 Side Training

A primary contributor to high activation memory usage is that retained activations are high-dimensional vectors produced by LLMs. To address this inefficiency, side-tuning (Zhang et al., 2020) freezes the LLM and trains a small-dimensional side network, resulting in lower-dimensional retained activations. Building on this idea, LST (Sung et al., 2022) adds an adaptation layer to each block of the LLM, connecting the original LLM with side networks. QST (Zhang et al., 2024b) further refines this concept by quantizing the original LLM into fewer bits while retaining higher precision in the side network.

In these methods, the side networks typically mirror the original LLM's architecture. However, several studies indicate that different architectures (e.g., CNNs, Adapters) can be effectively integrated into transformer-based LLMs, particularly for image processing. Notably, IISAN (Fu et al., 2024a) and UniPT (Diao et al., 2024b) explore the use of PET modules for side networks, and SHERL (Diao et al., 2024a) shows the effectiveness of late fusion network as side



(a) Parameter-efficient Training (b) Gradient Projection (c) Zeroth-order Training

Figure 3: **Illustrations of research for Optimizer**. (a) LoRA (Hu et al., 2021) for parameter-efficient training; (b) GaLore (Zhao et al., 2024) for gradient projection; (c) MeZO (Malladi et al., 2023) for zeroth-order training.

networks. Note that, since the side networks contain far fewer trainable parameters than the original LLMs, they can also benefit from reduced optimizer states, similar to PET (see §3.2.1).

3.1.3 Rematerialized Training

Instead of caching all activations during the forward pass, we can either cache only a subset of activations or skip caching entirely. For instance, gradient checkpointing (Chen et al., 2016) mitigates memory overhead by not storing intermediate activations in certain layers. However, these skipped activations must then be recomputed during backpropagation, increasing computational costs. Reversible networks (Liao et al., 2023b) similarly avoid saving activations. Compared to gradient checkpointing, reversible networks do not require additional forward passes to recompute activations, as they can directly derive the activations during backpropagation.

Instead of recomputing activations during the backward pass, several works avoid recomputation for specific components of LLMs (e.g., layers or sequences). TokenTune (Simoulin et al., 2024) bypasses intermediate activation retained at randomly selected token positions across all operations. Instead of the random selection, SAGE (Kim and Lee, 2025) selectively caches activations based on their saliency during the forward pass. DropBP (Woo et al., 2024) performs a standard forward pass in every layer but skips storing intermediate activations in randomly selected layers. Since aforementioned methods are typically orthogonal to other MEFT approaches, they are often integrated with complementary techniques, such as parameter-efficient fine-tuning.

3.1.4 Challenges

Reducing memory usage from activations is particularly beneficial when processing long sequences or using larger batch sizes. However, current methods still face several challenges. First, adaptive training approaches can introduce instability, as they bypass certain computations. In addition, the metrics used to modify computation—often heuristics based on attention maps or norms—tend to lack general applicability in diverse settings. Side training, on the other hand, involve additional backbone networks, resulting in more parameters and slightly higher memory consumption during inference (Sung et al., 2022). Moreover, because these methods rely only on original features without the task adaptation, they often underperform on challenging datasets (Sung et al., 2022; Zhang et al., 2024b). Finally, while rematerialized training indeed conserves memory, it often incurs higher computational costs due to the recomputation process.

3.2 Optimizer

The optimizer occupies a significant portion of training memory by maintaining state variables (e.g., momentum, variance) and gradients. These memory usages can be reduced by minimizing the number of trainable parameters, projecting gradients onto lower-dimensional subspaces, or avoiding backpropagation process entirely.

3.2.1 Parameter-efficient Training

The number of optimizer states maintained during training scales directly with the number of trainable parameters. Parameter-efficient training (PET) addresses this by restricting updates to only a subset of model parameters, which can be implemented via three modification types: **additive**, **selective**, or **reparameterization**.

Additive PET This approach augments the model with lightweight trainable modules while keeping the backbone parameters fixed. For example, adapter-based methods insert bottleneck layers—with nonlinear activation functions—into Transformer blocks, either along a serial path (Houlsby et al., 2019; Pfeiffer et al., 2021) or a parallel path (He et al., 2022a). Recently, to enhance the efficiency of adapters, several

studies have employed a sparse activation mechanism (Lei et al., 2023; He et al., 2022b) or utilized hypernetworks to generate task- and layer-specific adapter weights dynamically (Karimi Mahabadi et al., 2021).

Prompt-based methods (Lester et al., 2021; Li and Liang, 2021) steer LLMs by prepending learnable embeddings to activations (e.g., input embeddings or key, value representations in attention layers). For example, LLaMA-Adapter (Zhang et al., 2024a) prepends a set of learnable prompts controlled by a zero-initialized gating mechanism, allowing the model to adaptively regulate prompt embeddings. Otherwise, $(IA)^3$ (Liu et al., 2022) injects learnable prompts into both the attention and feed-forward layers, which learns to rescale the activations.

Selective PET This approach fine-tunes only a small subset of LLMs' parameters—leaving the rest frozen—rather than introducing additional modules. Therefore, identifying which parameters to update is a crucial design choice. For example, Lee et al. (2019) demonstrate that selectively fine-tuning a subset of layers can yield strong results, and Gheini et al. (2021) show that updating only the cross-attention weights suffices to achieve competitive translation performance. BitFit (Zaken et al., 2022) extends this concept by demonstrating that tuning only the bias terms can still yield competitive performance to full fine-tuning. To automate parameter selection, subsequent methods employ learnable criteria: Diff Pruning (Guo et al., 2021) learns a binary mask to identify tunable parameters; PaFi (Liao et al., 2023a) selects parameters with the smallest absolute magnitudes; and SAM (Fu et al., 2023) leverages a second-order approximation to refine this mask, enhancing both efficiency and performance.

Reparameterization-based PET This approach reconstructs the existing parameters with learnable low-rank adaptation weights. A representative method is LoRA (Hu et al., 2021), which introduces a bottleneck layer to approximate changes in the attention weights during fine-tuning. Subsequent work has enhanced LoRA's training stability, initialization schemes, and dynamic parameter utilization. For example, LoRA-FA (Zhang et al., 2023) further reduces both trainable parameters and activation memory by freezing LoRA's down-projection weights. DyLoRA (Valipour et al., 2023) introduces a dynamic search over the low-rank dimensions at each layer. DoRA (Liu et al., 2024a) decomposes the pre-trained weights into magnitude and directional components—better aligning with the capacity of full fine-tuning—and then applies LoRA-style low-rank updates to both components. MoLE (Wu et al., 2024b) learns to compose multiple pre-trained LoRA modules, enabling adaptation to diverse tasks and inputs. Likewise, MoDULA (Ma et al., 2024) combines a universal LoRA module with task-specific LoRA adapters for multi-task learning.

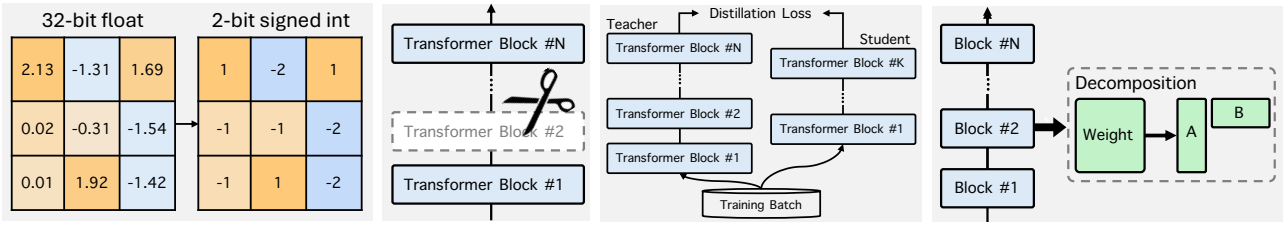
3.2.2 Gradient Projection

While effective, PET methods often underperform compared to full-rank weight training. This performance gap stems from constraining optimization to a low-dimensional subspace of the parameter space, which impairs training dynamics. To improve learning capacity while still reducing memory usage, GaLore (Zhao et al., 2024) applies a low-rank projection only to the gradients, enabling high-rank parameter updates with minimal memory overhead. Subsequent methods try to address GaLore's reliance on frequent SVD computations: Flora (Hao et al., 2024b) leverages random projections to compress gradients; RGP (Saheb Pasand and Bashivan, 2024) employs randomized SVD to preserve gradient-informed subspaces at lower cost; and Q-GaLore (Zhang et al., 2024c) introduces layer-wise adaptive subspace updates. These approaches collectively reduce computational burden while maintaining effective gradient updates. To further enhance memory efficiency, GRASS (Muhamed et al., 2024) introduces structured sparse projections, and GaLore-mini (Huang et al., 2024) groups parameters into blocks that share the same second-order optimizer state. Additionally, COAP (Xiao et al., 2024) introduces correlation-aware projections, and GradNormLoRP (Huang et al., 2025) incorporates weight normalization into the gradient projection process to improve training stability.

3.2.3 Zeroth-order Training

First-order optimization is often considered a bottleneck for memory efficiency. Zeroth-order (ZO) training offers a promising alternative by avoiding gradient backpropagation, thereby removing the need to store activations and optimizer states. At its core, ZO training estimates weight gradients via *simultaneous perturbation stochastic approximation* (Spall, 1992), which approximates gradients using only two forward passes per update. Consequently, ZO training requires memory resources similar to those used during inference.

A representative work is MeZO (Malladi et al., 2023), which adapts the classical ZO-SGD algorithm for in-place operation. However, ZO training still exhibits substantial performance gaps and training instability compared to first-order optimization (e.g., SGD, Adam). Therefore, the following works attempt to reduce such limitations. Specifically, AdaZeta



(a) Quantization (b) Pruning (c) Knowledge Distillation (d) Low-rank Factorization

Figure 4: **Illustrations of research for Model weights**. (a) RA-LoRA (Kim et al., 2024b) for quantization; (b) ShortGPT (Men et al., 2024) for pruning; (c) MiniLLM (Gu et al., 2024) for knowledge distillation; (d) FWSVD (Hsu et al., 2022) for low-rank factorization.

(Yang et al., 2024a) introduces fast forward tensorized adapters to address divergence issues from larger training parameters with an adaptive query schedule. Instead of training additional weights, Sparse-MeZO (Liu et al., 2024b) identifies a small set of salient parameters first and then optimizes these weights selectively using sparse masks. Several methods attempt to use additional information to more accurately estimate the gradients. MeZO-SVRG (Gautam et al., 2024) iteratively combines full-batch and mini-batch information to facilitate asymptotically unbiased and low-variance gradient estimators of ZO method. HiZOO (Zhao et al., 2025) estimates the diagonal Hessian using an additional forward pass to address heterogeneous curvatures across parameters. LOZO (Chen et al., 2025) tries to match the low-rank property of the first-order gradients by sampling the low-rank perturbation matrix.

3.2.4 Challenges

While PET methods leverage low-rank approximations to achieve efficiency, this property can be a double-edged sword—for example, leading to poor generalization in noisy environments (Kim et al., 2024d) or in federated learning settings (Babakniya et al., 2023). This observation underscores the need for more comprehensive studies across diverse environments. For gradient projection methods, the evaluation in most prior works primarily have focused on model scales up to 13B parameters. The effectiveness of these methods for significantly larger LLMs, exceeding hundreds of billions of parameters, requires further exploration. For ZO training methods, while they effectively reduce memory consumption, these methods still often suffer from slower convergence and lower model performance compared to first-order methods such as Adam optimizer. From a theoretical perspective, conducting a theoretical analysis of how zeroth-order gradient estimates regularize the performance of first-order methods is an important future research direction.

3.3 Model Weights

Large-scale model weights incur overall memory requirements for storing activations, weights, and optimizer states during training. These memory usages can be reduced by lowering numerical precision or minimizing the number of LLM parameters.

3.3.1 Quantization

Quantization is a technique to represent the weights from high-precision to lower-precision data. These techniques in model compression can be broadly categorized into two paradigms: **quantization-aware training** and **post-training quantization**.

Quantization-aware training (QAT) QAT techniques explicitly integrate quantization into the training phase. A prominent trend within QAT is to form hybrid methods by combining quantization with adapter-based approaches, particularly LoRA (see §3.2.1), to effectively preserve model performance. For example, QLoRA (Dettmers et al., 2023) finetunes 4-bit quantized LLMs with LoRA. PEQA (Kim et al., 2023a) finetunes only quantization scales per weight group in downstream tasks to avoid additional adapter parameters. LoftQ (Li et al., 2024b) jointly optimizes the quantization parameters and LoRA initialization. QA-LoRA (Xu et al., 2024) employs group-wise quantization during fine-tuning to effectively balance quantization and adaptation parameters. RA-LoRA (Kim et al., 2024b) dynamically adjusts adapter ranks to counteract quantization errors in extremely low-bit scenarios efficiently.

Post-training quantization (PTQ) PTQ compresses models after training, primarily aiming for efficient inference. Early PTQ methods like ZeroQuant (Yao et al., 2022a) and nuQmm (Park et al., 2022) use direct rounding of weights to the nearest quantization level. To minimize quantization errors, GPTQ (Frantar et al., 2022) introduces an approximate large-scale solver conducting layer-wise reconstruction, while AWQ (Lin et al., 2023), SpQR (Dettmers et al., 2024),

and OWQ (Lee et al., 2024a) isolate "outlier weight." SqueezeLLM (Kim et al., 2024c) introduces a non-uniform quantization to represent the distribution of weights better. Although these PTQ methods primarily target inference-time compression, their advanced quantization techniques, such as sophisticated rounding strategies and grouping methods, can directly benefit QAT frameworks (e.g., NF4 and group-wise quantization).

3.3.2 Pruning

Pruning (LeCun et al., 1989) is a powerful technique to reduce the size of the model by removing redundant parameters. Pruning methods can be categorized into **structured**, **semi-structured**, and **unstructured pruning**. However, we exclude unstructured pruning from the scope of this survey due to its reliance on specialized hardware and software support.

Structured pruning Structured pruning removes redundant components of LLMs in various granularities, such as layers, attention heads, and rows/columns of weights. For instance, ShortGPT (Men et al., 2024), Shortened LLaMA (Kim et al., 2024a), and SLEB (Song et al., 2024) identify and remove less impactful layers, while LaCo (Yang et al., 2024b) gradually merges similar layers from deep to shallow. Beyond layer-level pruning, LLM-Pruner (Ma et al., 2023) eliminates "coupled structures" (e.g., attention heads or row/column blocks) identified via gradient-based importance metrics, and FLAP (An et al., 2024) prunes entire columns using a fluctuation-based metric. Meanwhile, SliceGPT (Ashkboos et al., 2024) re-maps each layer's weights with a PCA-based orthogonal transformation, then slices off the least important dimensions.

Semi-structured pruning Despite those efforts, structured pruning is still overly coarse, often resulting in significant performance degradation. Therefore, semi-structured pruning methods such as SparseGPT (Frantar and Alistarh, 2023) and Wanda (Sun et al., 2024b) are proposed. Those methods adopt finer-grained sparsity patterns (e.g., 2:4 or 4:8), balancing hardware-friendly acceleration with more precise control over parameter pruning. MaskLLM (Fang et al., 2024) also advances semi-structured pruning by learning transferable N:M sparsity masks through end-to-end training with frozen weights.

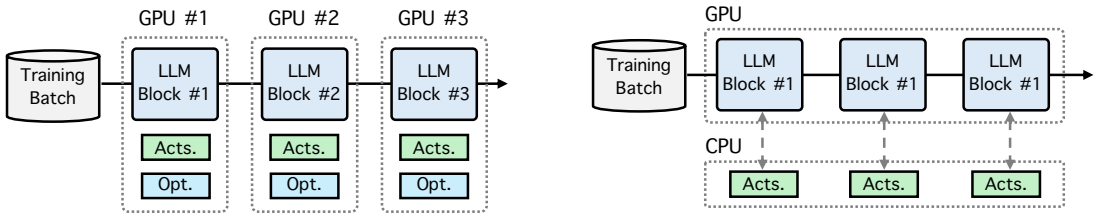
3.3.3 Knowledge Distillation

Knowledge distillation (Hinton et al., 2015) transfers knowledge from a larger teacher model to a smaller student model. We categorize methods into **text-based distillation**, which trains students using teacher-generated texts (e.g., chain-of-thought (CoT), instructions), and **representation-based distillation**, which utilizes teacher's internal representations (e.g., logits, hidden activations), thus limited to open-source models.

Text-based distillation To compensate for the inaccessibility of closed-source LLMs, prior works leverage their textual outputs to distill knowledge. One major stream is chain-of-thought distillation, where student models enhance their reasoning ability by learning reasoning traces generated by LLMs (Li et al., 2022; Ho et al., 2023; Magister et al., 2023; Hsieh et al., 2023). To advance this paradigm, Socratic CoT (Shridhar et al., 2023) decomposes complex questions into sub-questions, SCOTT (Wang et al., 2023a) enforces rationale-answer consistency, and PAD (Zhu et al., 2024a) incorporates structured reasoning programs. Another stream of work is instruction following distillation, where students learn with instruction-response pairs generated from LLMs (Wang et al., 2023b; Peng et al., 2023; Wu et al., 2024a). It aims to improve the zero-shot capability of small-language models. Meanwhile, recent works (Li et al., 2023, 2024a) aim to improve textual outputs through reflection-based self-improvement and student-guided selection.

Representation-based distillation Initially, it was widely used to compress pre-trained language models. Early methods such as DistillBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2020), and MobileBERT (Sun et al., 2020) distilled soft labels, hidden states, and attention distributions, while later approaches like MiniLM (Wang et al., 2020) and MiniLMv2 (Wang et al., 2021) focused solely on attention-based distillation for greater architectural flexibility. Tutor-KD (Kim et al., 2022) extends this line by generating training examples that are easy for the teacher but challenging for the student, enhancing the quality of transferred representations.

However, due to the dominance of closed-source LLMs, this line of research had stagnated for a while. Recently, with the release of open-source models like LLaMA (Dubey et al., 2024), momentum has been regained. For instance, MiniLLM (Gu et al., 2024) introduces reverse Kullback-Leibler divergence to handle the complexity of LLM output spaces. In parallel, Mentor-KD (Lee et al., 2024b) introduces an intermediate mentor model to bridge the gap between closed-source LLMs and student models, enabling richer representation-based distillation even without access to the teacher's internal states.



(a) Distributed Training (b) Memory Offloading

Figure 5: **Illustrations of research for System-level Optimization**. (a) Pipeline parallel (Huang et al., 2019) for distributed training; (b) CPU Offloading of activations for memory offloading.

3.3.4 Low-rank Factorization

Another practical way to compress LLMs is by factorizing large-weight matrices into low-rank forms, typically via Singular Value Decomposition (SVD). However, naive SVD can yield large compression errors when truncating singular values crucial to model performance. To address this, FWSVD (Hsu et al., 2022) addresses the mismatch between SVD's reconstruction objective and task accuracy by incorporating Fisher information to weight parameter importance, while ASVD (Yuan et al., 2023) manages activation outliers and adaptively assigns ranks across layers. Meanwhile, SVD-LLM (Wang et al., 2025) introduces a truncation-aware whitening step with sequential low-rank updates to address SVD's misalignment with compression loss and the lack of post-truncation parameter tuning, while MoDeGPT (Lin et al., 2025) applies joint decomposition across sub-modules without additional adapters.

3.3.5 Challenges

Compressing the model's weights can significantly reduce the overall memory requirements during training, but each compression approach introduces distinct challenges. Quantization requires frequent dequantization (up-scaling) steps to accommodate high-precision computations, thereby increasing computational costs and latency. Pruning generally incurs substantial accuracy loss, and although semi-structured pruning can partially mitigate this issue, its effectiveness remains limited to specific hardware support. Knowledge distillation poses the challenge of selecting an appropriate student model, as its size and architecture must be determined empirically. There is no guarantee that a given student can effectively absorb the teacher's knowledge, and multiple rounds of distillation with different configurations are often required. Lastly, low-rank factorization necessitates aggressive rank reductions to achieve meaningful compression, thereby inevitably leading to slower convergence or performance degradation.

4 System-level Optimization

4.1 Distributed Training

Fine-tuning recent LLMs typically requires substantial memory, often exceeding the capacity of a single GPU. This limitation has motivated the development of distributed training strategies that parallelize computation along different dimensions (e.g., data, layers, or sequences) to reduce the memory burden on individual devices. To further enhance scalability and efficiency, these strategies are often integrated into hybrid approaches.

Data Parallelism (DP) This approach replicates the entire model across N devices, with each device processing a different micro-batch of data. While standard DP implementations (e.g., PyTorch DDP) are relatively straightforward, they incur high memory redundancy since each device must store a complete copy of the model's parameters, gradients, and optimizer states. To mitigate this issue, memory-efficient variants such as ZeRO (Rajbhandari et al., 2020) and its native PyTorch implementation, FSDP (Zhao et al., 2023), have been introduced. These methods shard the model states (parameters, gradients, and optimizer states) across devices, thereby reducing the per-GPU memory footprint to roughly 1/N of the original and enabling the training of substantially larger models.

Tensor Parallelism (TP) This approach partitions the model within individual layers, also known as intra-layer (tensor) parallelism. Large matrix multiplications are split across multiple devices, with each GPU storing only a slice of the weight matrices and activations. This distributes both the computational load and the memory required for parameters. However, it also introduces substantial communication overhead, typically necessitating an all-reduce operation to synchronize results after the parallel computation. This technique was first popularized by Megatron-LM (Shoeybi et al., 2019) and later optimized to overlap computation with communica-

tion, enabling near-linear scaling to trillion-parameter models (Narayanan et al., 2021).

Pipeline Parallelism (PP) This approach partitions the model across layers, also known as inter-layer (pipeline) parallelism. The model's layers are grouped into stages, with each stage assigned to a different device. A data batch is divided into micro-batches, which are passed through the pipeline of stages to keep all devices utilized. This substantially reduces per-GPU memory requirements, as each device only stores the parameters of its assigned layers. Efficient scheduling—such as the micro-batching strategy in GPipe (Huang et al., 2019)—is critical to minimizing pipeline idle time, while alternative schemes such as 1F1B (Narayanan et al., 2019) offer a trade-off between idle time and weight staleness.

Sequence Parallelism (SP) For long sequences, activation memory—particularly in attention layers—can become a critical bottleneck. SP alleviates this by partitioning the input data along the sequence-length, thereby distributing the activations across devices. To reconstruct the full activations when needed, the approach relies on communication patterns such as all-to-all or ring-based exchanges. Representative methods include Ulysses (Jacobs et al., 2024) and Ring Attention (Liu et al., 2023).

4.1.1 Challenges

Distributed training can enlarge the feasible scale of fine-tuning by distributing the memory and computation load across multiple devices, but each strategy has a challenge that remains unsolved. Data parallelism, even with a sharded optimizer, still incur high memory for activations and communication overhead because devices must gather full parameters and gradients during forward/backward passes. Tensor parallelism divides matrices across devices, but it requires storing complete intermediate activations and performing multiple all-reduce operations per layer. Although multi-dimensional partitioning reduces this cost, communication overhead and activation storage remain bottlenecks. Pipeline parallelism cuts per-device parameter memory by splitting layers into stages. Micro-batching creates pipeline bubbles (or, under 1F1B, reduces bubbles at the cost of possible weight staleness), and each stage still has to keep activations until the backward pass, so overall memory savings are limited. Sequence parallelism alleviates activation memory for long inputs, but it relies on expensive all-to-all communication to reconstruct full sequences and still has a high peak memory footprint even with optimized ring-attention and selective recomputation. Moreover, none of these approaches reduce the inherent memory required for fine-tuning, so they cannot support real-world on-device training without combining with other MEFT methods.

4.2 Memory Offloading

Since a large portion of memory during training is consumed by temporarily retained activations and optimizer states for backpropagation, storing these tensors on expensive high-computing hardware leads to inefficient resource utilization. This inefficiency has driven research into memory offloading, a technique that moves these transient tensors to larger-capacity resources like CPU main memory or lower-cost storage, such as an NVMe SSD.

The effectiveness of memory offloading hinges on the swapping policy between different devices. Instead of designing the policy in heuristic manner (Rhu et al., 2016; Le et al., 2018), SwapAdvisor (Huang et al., 2020) introduces an optimization-based swapping system to support various kinds of large model training. Specifically, given the computational graphs of the large model, it plans for what and when to swap prior to execution through genetic algorithms. Sparse Adapter (Hao et al., 2024a) integrates an offloading policy into the parallel adapter (He et al., 2021)(see §3.2.1), such that only the necessary activations for the current inputs are retrieved (i.e., swapped) from the CPUs.

The principles of distributed training and memory offloading are not mutually exclusive; in fact, several approaches integrate both. A prominent example is ZeRO-Offload (Ren et al., 2021), a hybrid method that extends the ZeRO sharding strategy (see §4.1), which complements data parallelism by partitioning optimizer states and update computations to the larger CPU memory pools while keeping the forward and backward passes on GPUs. Similarly, ZeRO-Infinity (Rajbhandari et al., 2021) integrates 3D parallelism (i.e., tensor, pipeline, data parallelism) with a heterogeneous offloading system which exploits CPUs and NVMe memory simultaneously.

4.2.1 Challenges

Offloading transient data (e.g., activations and optimizer states) to lower-cost devices enables more efficient utilization of GPU resources. However, since this approach requires frequent communication between heterogeneous devices, it can introduce latency and cause GPUs to remain idle, depending on the scheduling design. Moreover, variations in LLM architectures, computational mechanisms, and system configurations demand carefully optimized offloading and communication strategies tailored to each environment. Therefore, striking the right balance be-

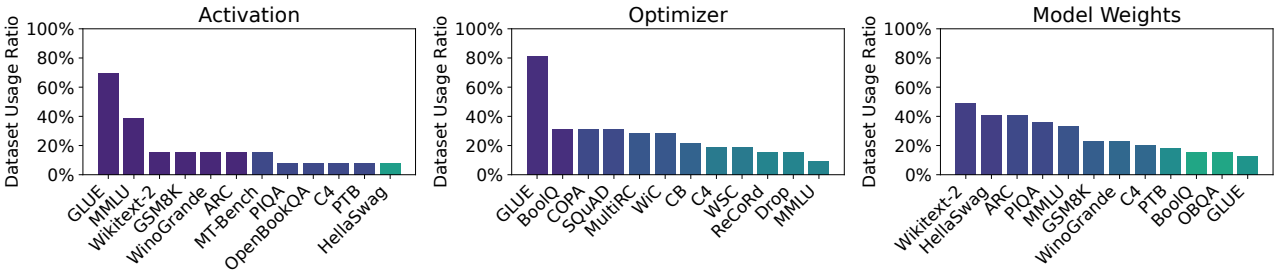


Figure 6: **Usage ratio of MEFT benchmarks**. We report the top 12 benchmarks in each of the three categories—activations, optimizer, and model weights—collected from the 84 papers surveyed in this work.

tween communication overhead and GPU memory usage is critical, as it can significantly impact overall system performance.

5 Metrics and Benchmark Datasets

Before presenting the empirical analysis of representative methods, we discuss the evaluation metrics and benchmarks used to assess MEFT approaches.

5.1 Metrics

Memory efficiency in fine-tuning LLMs can be evaluated using two primary metrics, peak memory usage and average memory usage. These metrics are commonly presented alongside training time, computational costs, and task-specific performance to assess the MEFT methods comprehensively.

Peak Memory Usage. Peak memory usage refers to the maximum amount of memory utilized during the training process. It serves as a critical indicator of worst-case memory demands, thereby ensuring that proposed methods can operate on hardware with limited resources. Excessively high peak memory usage may result in out-of-memory (OOM) errors during the most resource-intensive segments of training.

Average Memory Usage. Average memory usage represents the mean amount of memory consumed during the entire training process. While it is not the primary metric for avoiding OOM errors—since it depends more directly on peak memory usage—it provides valuable insight into the overall resource consumption profile. In cloud-based environments, lower average memory usage can lead to reduced operational costs, improved hardware stability, and potentially more efficient scheduling of concurrent tasks.

Convergence Speed and Computational Costs. Current MEFT methods effectively reduce memory usage during fine-tuning, but this efficiency often comes at the expense of slower convergence and increased computational overhead. Therefore, it is essential to evaluate practical efficiency holistically, considering metrics beyond memory consumption. Convergence speed is typically measured by wall-clock training time and by comparing loss trajectories against a baseline model, while computational cost is often quantified in terms of floating-point operations (FLOPs). By examining these metrics alongside memory usage, researchers can determine whether a given technique genuinely improves overall efficiency or merely shifts the bottleneck.

Task-specific Performance. Efficiency is typically formalized as the cost of a model relative to the results it produces (Schwartz et al., 2020). Preserving task-specific performance of the original models is therefore crucial. Researchers typically report task-specific metrics (e.g., ROUGE, perplexity, accuracy) to assess whether MEFT methods adversely affect model performance or to demonstrate the trade-off between costs and task-specific performance.

5.2 Benchmark Datasets

Benchmark datasets are essential for evaluating MEFT methods, providing standardized references to test generalization across scenarios. To capture current practices, we gather all benchmarks from the surveyed papers and analyze their usage frequencies.

Figure 6 presents the usage distributions. One notable observation is the marked imbalance in evaluation practices for activation- and optimizer-focused methods. Over 70% of studies evaluate their efficient methods on the GLUE benchmark (Wang et al., 2019). Although GLUE is valued for its diverse task suite, such heavy reliance raises concerns about whether proposed MEFT methods generalize beyond relatively simple linguistic scenarios. Beyond GLUE, recent works evaluate on commonsense reasoning benchmarks such as PIQA (Bisk et al., 2020), CommonsenseQA (Talmor et al., 2019), ARC (Clark et al., 2018), and WinoGrande (Sakaguchi et al., 2021), as well as on math problem datasets like GSM8K (Cobbe et al., 2021). Moreover, to test the generalization capabilities of fine-tuned LLMs, several stud-

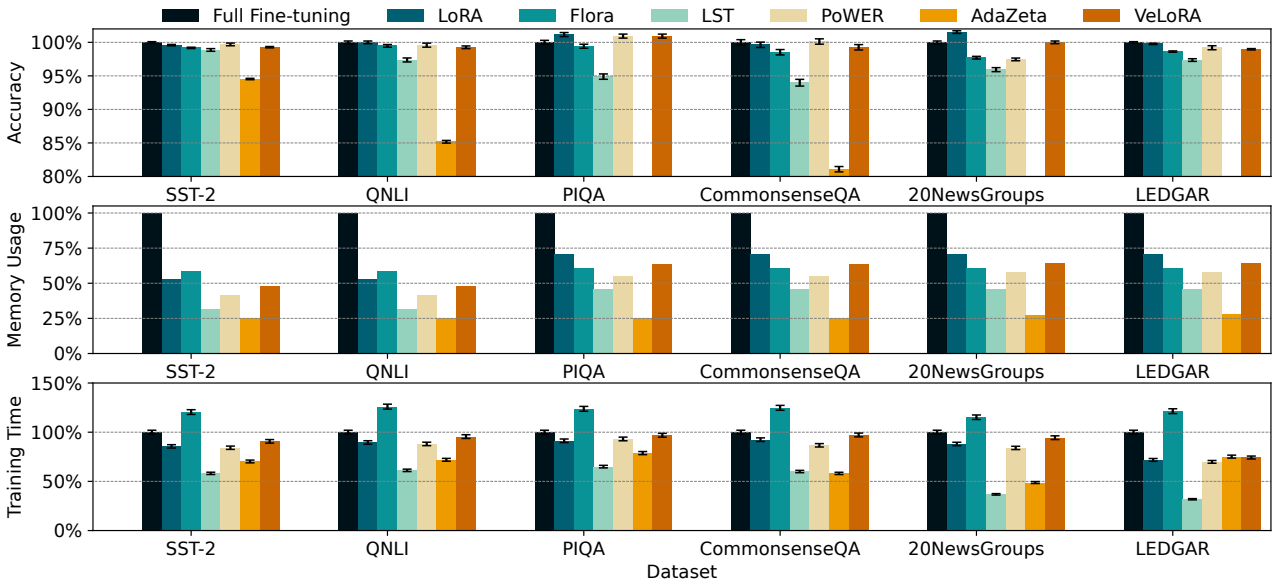


Figure 7: **Investigation of MEFT Methods.** The reported values for accuracy, peak memory, and training time represent relative changes compared to full fine-tuning. Here, we used Llama-3.2-3B-Instruct as the backbone.

ies train on instruction-following corpora (e.g., Alpaca (Taori et al., 2023), OpenAssistant Conversations (Köpf et al., 2023)) and then measure zero-shot performance on MMLU (Hendrycks et al., 2021), BigBench-Hard (Suzgun et al., 2023), or conversational benchmarks such as MT-Bench (Zheng et al., 2023), using LLM-based evaluation.

6 Empirical Survey

In this section, we empirically analyze the representative methods for MEFT.

6.1 Experimental Setups

Methods In this analysis, we focus on model-level optimizations, as system-level methods can be applied orthogonally across different approaches. Specifically, we compare methods aimed at reducing memory associated with *activations* and *optimizers*, since weight-compression techniques can be seamlessly integrated with existing approaches and have been thoroughly analyzed in a prior survey (Zhu et al., 2024b). To select appropriate methods for our experiments, we considered three key aspects: i) **Reproducibility**: The method must have an official implementation, and its codebase should be easy to integrate without requiring significant modifications. ii) **Representativeness**: The method should be well-recognized within its category (e.g., seminal or widely cited work), serving as a representative of its research direction. iii) **Practicality**: The method should avoid overly complex components and be broadly applicable across various tasks and domains. Based on these criteria, we compare the following methods: **LoRA** (Hu et al., 2021), **Flora** (Hao et al., 2024b), and **AdaZeta** (Yang et al., 2024a) which aim to reduce memory associated with the optimizers; **LST** (Sung et al., 2022), **PoWER** (Goyal et al., 2020), and **VeLoRA** (Miles et al., 2024) which aim to reduce memory associated with the activations. The implementation details (e.g., hyper-parameters, maximum sequence lengths, model architectures) are described in Appendix A.

Datasets To assess the general applicability of surveyed methods, we conduct experiments on diverse benchmarks: SST-2 (Socher et al., 2013) and QNLI (Rajpurkar et al., 2016) from GLUE (classification); PIQA and CommonsenseQA for commonsense reasoning (multiple-choice question answering); 20NewsGroups (Lang, 1995) and LEDGAR (Tuggener et al., 2020) for document understanding.

Evaluation Metrics We evaluate each method across three dimensions: task accuracy, peak GPU memory usage, and training time. Peak memory usage is measured using `nvidia-smi` to measure practical usage, and wall-clock time is used to measure the training time. All results are normalized by computing their relative change from full fine-tuning.

6.2 Evaluation Results

Figure 7 presents comparison results from simple sentence classification to document-level understanding. From this analysis, we observe that almost all methods achieve strong performance on GLUE tasks (SST-2, QNLI), preserving accuracy within 5% of full fine-tuning. Importantly, however, several methods perform poorly on challenging benchmarks. Side training method (LST) suffers a signifi-

cant accuracy drop relative in commonsense reasoning benchmarks, i.e., more than 5% loss in accuracy. Specifically, while zeroth-order training (AdaZeta) shows promising results in terms of memory usages, it fails to achieve competitive accuracy beyond the GLUE benchmarks—a limitation that has also been noted in several studies (Malladi et al., 2023). This suggests that evaluating methods solely on GLUE tasks—the most frequently used evaluation benchmark (Figure 6)—risks overestimating their generalization ability. Nevertheless, since LST delivers substantial gains in memory efficiency—and even in training time—using it when resource constraints outweigh accuracy considerations can be advantageous. Interestingly, the adaptive training method (PoWER), which is not originally designed for memory efficiency, performs well across diverse benchmarks—achieving reduced training time and significant memory savings. These results suggest that exploring more diverse adaptive methods is a promising direction for MEFT. Overall, methods that reduce activation memory (LST, PoWER) tend to incur lower resource costs but risk decreased accuracy, whereas methods targeting memory associated with the optimizers often match full fine-tuning accuracy at the expense of higher memory or time overhead than activation research. Consequently, choosing methods tailored to specific resource environments and situations (accuracy vs. memory and time) is advisable.

7 Discussion and Future Research

7.1 Further Exploration on Adaptive Methods

While adaptive training is originally designed for accelerating inference rather than MEFT, they exhibit promising results in improving memory efficiency compared to existing MEFT approaches. Therefore, exploring more advanced and diverse adaptive methods represents a promising direction for achieving a better trade-off between accuracy and resource costs.

7.2 Evaluation on Diverse Environments

While existing methods have empirically demonstrated memory efficiency, most evaluations are conducted on the GLUE benchmark, which primarily comprises relatively simple tasks rather than complex reasoning challenges. We also show that strong performance on GLUE does not always generalize to other tasks. To validate the broad efficiency of MEFT methods, it is important to explore diverse experimental setups (e.g., diverse sequence length, noisy label environments) and more challenging datasets (e.g., mathematical reasoning, commonsense reasoning). Moreover, recent studies have revealed that efficient methods exhibit distinct limitations in learning capacity—such as handling noisy labels (Kim et al., 2024d), heterogeneous federated settings (Babakniya et al., 2023)—underscoring the need to investigate diverse data distributions.

7.3 Evaluation Metrics and Trade-offs

Evaluating the practical efficiency of MEFT methods is always challenging, as it involves multiple considerations beyond memory savings alone. However, many current research papers provide only a partial view or focus solely on theoretical efficiency, which can lead to suboptimal performance and unexpected overheads in real-world applications. Therefore, it is essential to assess each method across multiple dimensions—at a minimum, training time, peak GPU memory usage, and task accuracy—to obtain a comprehensive evaluation.

8 Conclusion

In this survey, we present the first comprehensive survey dedicated to MEFT for LLMs. By categorizing existing methods according to optimization environments and key components that significantly contribute to memory usage, we provide a structured understanding of the design space. We also discuss current benchmarking strategies and evaluation metrics, highlighting their limitations and challenges. Our empirical comparisons further reveal the trade-offs associated with each approach, offering practical insights for real-world applications. Through this systematic investigation, we aim to provide a valuable and accessible reference that helps researchers and practitioners better understand the current landscape and encourages continued exploration in the MEFT of LLMs.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2025-00517221 and No.RS-2024-00415812) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00439328, Karma: Towards Knowledge Augmentation for Complex Reasoning (SW Starlab), No.RS-2024-00457882, AI Research Hub Project, and No.RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University)).

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10865–10873. AAAI Press.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*. OpenReview.net.

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, MOSTAFA EL-Khamy, and Salman Avestimehr. 2023. Slora: Federated parameter efficient fine-tuning of language models. In *International Workshop on Federated Learning in the Age of Foundation Models*.

Gantavya Bhatt, Yifang Chen, Arnav Das, Jifan Zhang, Sang Truong, Stephen Mussmann, Yinglun Zhu, Jeff Bilmes, Simon Du, Kevin Jamieson, et al. 2024. An experimental design framework for label-efficient supervised finetuning of large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6549–6560. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on Artificial Intelligence*, pages 7432–7439. AAAI Press.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Yiming Chen, yuan zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. 2025. Enhancing zeroth-order fine-tuning for language models with low-rank structures. In *International Conference on Learning Representations*. OpenReview.net.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. Spqr: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*. OpenReview.net.

Haiwen Diao, Bo Wan, Xu Jia, Yunzhi Zhuge, Ying Zhang, Huchuan Lu, and Long Chen. 2024a. SHERL: synthesizing high accuracy and efficient memory for resource-limited transfer learning. In *Proceedings of the European Conference on Computer Vision*, volume 15102, pages 75–95. Springer.

Haiwen Diao, Bo Wan, Ying Zhang, Xu Jia, Huchuan Lu, and Long Chen. 2024b. Unipt: Universal parallel tuning for transfer learning with efficient parameter and memory. In *Conference on Computer Vision and Pattern Recognition*, pages 28729–28740. IEEE.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*. OpenReview.net.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. 2024. Maskllm: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 7736–7758. Curran Associates, Inc.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323.

Junchen Fu, Xuri Ge, Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, Jie Wang, and Joemon M Jose. 2024a. Iisan: Efficiently adapting multimodal representation for sequential recommendation with decoupled peft. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 687–697.

Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. 2024b. Lazyllm: Dynamic token pruning for efficient long context llm inference. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.

Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12799–12807. AAAI Press.

Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. 2024. Variance-reduced zeroth-order methods for fine-tuning language models. In *International Conference on Machine Learning*, pages 15180–15208.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765. Association for Computational Linguistics.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *Proceedings of the International Conference on Machine Learning*, pages 3690–3699.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*. OpenReview.net.

Yue Guan, Zhengyi Li, Jingwen Leng, Zhouhan Lin, and Minyi Guo. 2022. Transkimmer: Transformer learns to layer-wise skim. *arXiv preprint arXiv:2205.07324*.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4884–4896. Association for Computational Linguistics.

Jitai Hao, WeiWei Sun, Xin Xin, Qi Meng, Zhumin Chen, Pengjie Ren, and Zhaochun Ren. 2024a. Meft: Memory-efficient fine-tuning through sparse adapter. *arXiv preprint arXiv:2406.04984*.

Yongchang Hao, Yanshuai Cao, and Lili Mou. 2024b. Flora: Low-rank adapters are secretly gradient compressors. In *International Conference on Machine Learning*, pages 17554–17571.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022a. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*. OpenReview.net.

Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. 2022b. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2184–2190. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 14852–14882. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017. Association for Computational Linguistics.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*. OpenReview.net.

Chien-Chin Huang, Gu Jin, and Jinyang Li. 2020. Swapadvisor: Pushing deep learning beyond the gpu memory limit via smart swapping. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1341–1355.

Jia-Hong Huang, Yixian Shen, Hongyi Zhu, Stevan Rudinac, and Evangelos Kanoulas. 2025. Gradient weight-normalized low-rank projection for efficient llm training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 24123–24131. AAAI Press.

Weihao Huang, Zhenyu Zhang, Yushun Zhang, Zhi-Quan Luo, Ruoyu Sun, and Zhangyang Wang. 2024. Galore-mini: Low rank gradient learning with fewer learning rates. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Reza Yazdani Aminadabi, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. 2024. System optimizations for enabling training of extreme long sequence transformer models. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 121–130.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 565–576. Association for Computational Linguistics.

Bo-Kyeong Kim, Geon-min Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024a. Shortened llama: A simple depth pruning for large language models. *CoRR*.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2023a. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. volume 36, pages 36187–36207. Curran Associates, Inc.

Junho Kim, Jun-Hyung Park, Mingyu Lee, Wing-Lam Mok, Joon-Young Choi, and SangKeun Lee. 2022. Tutoring helps students learn better: Improving knowledge distillation for BERT with tutor

network. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 7371–7382. Association for Computational Linguistics.

Minsoo Kim, Sihwa Lee, Wonyong Sung, and Jungwook Choi. 2024b. RA-LoRA: Rank-adaptive parameter-efficient fine-tuning for accurate 2-bit quantized large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15773–15786. Association for Computational Linguistics.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. 2024c. Squeezellm: Dense-and-sparse quantization. In *International Conference on Machine Learning*, pages 23901–23923.

Yeachan Kim, Junho Kim, and SangKeun Lee. 2024d. Towards robust and generalized parameter-efficient fine-tuning for noisy label learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 5922–5936. Association for Computational Linguistics.

Yeachan Kim, Junho Kim, Jun-Hyung Park, Mingyu Lee, and SangKeun Lee. 2023b. Leap-of-thought: Accelerating transformers via dynamic token routing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 15757–15769. Association for Computational Linguistics.

Yeachan Kim and SangKeun Lee. 2025. Forward knows efficient backward path: Saliency-guided memory-efficient fine-tuning of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9341–9356.

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36:47669–47681.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, pages 331–339. Elsevier.

Tung D Le, Haruki Imai, Yasushi Negishi, and Kiyokuni Kawachiya. 2018. Tflms: Large model support in tensorflow by graph rewriting. *arXiv preprint arXiv:1807.02037*.

Yann LeCun, John S. Denker, and Sara A. Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan Kaufmann.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024a. OWQ: outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13355–13364. AAAI Press.

Hojae Lee, Junho Kim, and SangKeun Lee. 2024b. Mentor-KD: Making small language models better multi-step reasoners. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 17643–17658. Association for Computational Linguistics.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*.

Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, et al. 2023. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172.

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

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024a. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16189–16211. Association for Computational Linguistics.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Heng Huang, Jiuxiang Gu, and Tianyi Zhou. 2023. Reflection-tuning: Data recycling improves llm instruction-tuning. *arXiv preprint arXiv:2310.11716*.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing

Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022. Explanations from large language models make small reasoners better. *CoRR*, abs/2210.06726.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597. Association for Computational Linguistics.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2024b. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*. OpenReview.net.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Baohao Liao, Yan Meng, and Christof Monz. 2023a. Parameter-efficient fine-tuning without introducing new latency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 4242–4260. Association for Computational Linguistics.

Baohao Liao, Shaomu Tan, and Christof Monz. 2023b. Make pre-trained model reversible: From parameter to memory efficient fine-tuning. *Advances in Neural Information Processing Systems*, 36:15186–15209.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2025. ModeGPT: Modular decomposition for large language model compression. In *International Conference on Learning Representations*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. AWQ: activation-aware weight quantization for LLM compression and acceleration. *CoRR*, abs/2306.00978.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, pages 32100–32121.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. 2024b. Sparse mezo: Less parameters for better performance in zeroth-order LLM fine-tuning. *CoRR*, abs/2402.15751.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc.

Yufei Ma, Zihan Liang, Huangyu Dai, Ben Chen, Dehong Gao, Zhuoran Ran, Wang Zihan, Linbo Jin, Wen Jiang, Guannan Zhang, et al. 2024. Modula: Mixture of domain-specific and universal lora for multi-task learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2758–2770.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adámek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1773–1781. Association for Computational Linguistics.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large

language models are more redundant than you expect. *CoRR*, abs/2403.03853.

Roy Miles, Pradyumna Reddy, Ismail Elezi, and Jiankang Deng. 2024. Velora: Memory efficient training using rank-1 sub-token projections. *Advances in Neural Information Processing Systems*, 37:42292–42310.

Aashiq Muhamed, Oscar Li, David P. Woodruff, Mona Diab, and Virginia Smith. 2024. GRASS: compute efficient low-memory LLM training with structured sparse gradients. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 14978–15003. Association for Computational Linguistics.

Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. Pipedream: generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 1–15. ACM.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient large-scale language model training on GPU clusters using megatron-lm. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, page 58. ACM.

Dang Nguyen, Wenhan Yang, Rathul Anand, Yu Yang, and Baharan Mirzasoleiman. 2025. Mini-batch coresets for memory-efficient language model training on data mixtures. In *International Conference on Learning Representations*. OpenReview.net.

Rui Pan, Xiang Liu, SHIZHE DIAO, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, pages 57018–57049. Curran Associates, Inc.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2022. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. *CoRR*, abs/2304.03277.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 487–503. Association for Computational Linguistics.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564.

Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Stephen W Keckler. 2016. vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13. IEEE.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. 2024. Compressing large language models using low rank and low precision decomposition. In *Advances in Neural Information Processing Systems*, volume 37, pages 88981–89018. Curran Associates, Inc.

Ali Saheb Pasand and Pouya Bashivan. 2024. RGP: Achieving memory-efficient model fine-tuning via randomized gradient projection. In *Proceedings of The 4th NeurIPS Efficient Natural Language and Speech Processing Workshop*, pages 47–54.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63(12):54–63.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*. OpenReview.net.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073. Association for Computational Linguistics.

Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. 2024. Memory-efficient fine-tuning of transformers via token selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 21565–21580. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1631–1642. Association for Computational Linguistics.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: streamlining llms through redundancy verification and elimination of transformer blocks. In *International Conference on Machine Learning*, pages 46136–46155.

James C Spall. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024a. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*. OpenReview.net.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024b. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170. Association for Computational Linguistics.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. volume 35, pages 12991–13005. Curran Associates, Inc.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4149–4158. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Marcos Treviso, Ji-Ung Lee, Tianchu Ji, Betty van Aken, Qingqing Cao, Manuel R Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, et al. 2023. Efficient methods for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

Don Tuggener, Pius Von Däniken, Thomas Peetz, and Mark Cieliebak. 2020. Ledgar: A large-scale multi-label corpus for text classification of legal provisions in contracts. In *Proceedings of the Language Resources and Evaluation Conference*, pages 1235–1241. Association for Computational Linguistics.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. 2024. Efficient large language models: A survey. *Transactions on Machine Learning Research*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023a. SCOTT: Self-consistent chain-of-thought distillation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 5546–5558. Association for Computational Linguistics.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates Inc.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*. OpenReview.net.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 13484–13508. Association for Computational Linguistics.

Sunghyeon Woo, Baeseong Park, Byeongwook Kim, Minjung Jo, Se Jung Kwon, Dongsuk Jeon, and Dongsoo Lee. 2024. Dropbp: Accelerating fine-tuning of large language models by dropping backward propagation. In *Advances in Neural Information Processing Systems*, pages 20170–20197. Curran Associates, Inc.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2024a. Lamini-lm: A diverse herd of distilled models from large-scale instructions. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 944–964. Association for Computational Linguistics.

Xun Wu, Shaohan Huang, and Furu Wei. 2024b. Mixture of lora experts. In *The Twelfth International Conference on Learning Representations*.

Jinqi Xiao, Shen Sang, Tiancheng Zhi, Jing Liu, Qing Yan, Linjie Luo, and Bo Yuan. 2024. COAP: memory-efficient training with correlation-aware gradient projection. *CoRR*, abs/2412.00071.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2024. QA-loRA: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*. OpenReview.net.

Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. 2024a. Adazeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In *Proceedings of the Conference*

on Empirical Methods in Natural Language Processing, pages 977–995. Association for Computational Linguistics.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024b. Laco: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417. Association for Computational Linguistics.

Yu Yang, Hao Kang, and Baharan Mirzasoleiman. 2023. Towards sustainable learning: Coresets for data-efficient deep learning. In *International Conference on Machine Learning*, pages 39314–39330.

Kai Yao, Penglei Gao, Lichun Li, Yuan Zhao, Xiaofeng Wang, Wei Wang, and Jianke Zhu. 2024. Layer-wise importance matters: Less memory for better performance in parameter-efficient fine-tuning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1977–1992.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022a. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems*, volume 35, pages 27168–27183. Curran Associates, Inc.

Zhewei Yao, Xiaoxia Wu, Conglong Li, Connor Holmes, Minjia Zhang, Cheng Li, and Yuxiong He. 2022b. Random-ltd: Random and layerwise token dropping brings efficient training for large-scale transformers. *CoRR*, abs/2211.11586.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023. ASVD: activation-aware singular value decomposition for compressing large language models. *CoRR*, abs/2312.05821.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020. Side-tuning: A baseline for network adaptation via additive side networks. In *Proceedings of the European Conference on Computer Vision*, volume 15102, pages 698–714. Springer.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*.

Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. 2024a. Llama-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *International Conference on Learning Representations*.

Zhengxin Zhang, Dan Zhao, Xupeng Miao, Gabriele Oliaro, Qing Li, Yong Jiang, and Zhihao Jia. 2024b. Quantized side tuning: Fast and memory-efficient tuning of quantized large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1–17. Association for Computational Linguistics.

Zhenyu Zhang, Ajay Jaiswal, Lu Yin, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. 2024c. Q-galore: Quantized galore with INT4 projection and layer-adaptive low-rank gradients. *CoRR*, abs/2407.08296.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. In *Proceedings of the International Conference on Machine Learning*, pages 61121–61143.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. 2025. Second-order fine-tuning without pain for LLMs: A hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*. OpenReview.net.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xinwei Long, Zhouhan Lin, and Bowen Zhou. 2024a. Pad: Program-aided distillation can teach small models reasoning better than chain-of-thought fine-tuning.

In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2571–2597. Association for Computational Linguistics.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024b. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

Appendix

A Training Details on Empirical Survey

Here, we describe the implementation details of the methods used in the empirical analysis (Section 6). The hyperparameters for each method are summarized in Table 1. Since PoWER was not originally designed for efficient training, we apply LoRA (Hu et al., 2021) to enhance overall memory efficiency instead of performing full fine-tuning. In addition, within this method, the upper- and lower-bound ratios are determined based on the maximum sequence length, which automatically specifies the number of tokens in each layer. For each method, we adopt the hyperparameter settings from the original papers and their official code implementations. All experiments are conducted on NVIDIA RTX A6000 GPUs.

Table 1: **Training Setups**. Common and method-specific hyper-parameters used in empirical analysis.

Method	Hyper-parameters	SST-2	QNLI	PIQA	CSQA	20NewsGroups	LEDGAR
Common	Data Type	bfloat16					
	Batch size	16	16	16	16	8	8
	Epochs	3		6		3	
	warm-up ratio	0.06					
	Seq. Length	128		256		512	
	Model	Llama-3.2-3B-Instruct					
Full Fine-tuning	Learning rate	2e-5					
	Optimizer	AdamW					
	LR Schedule	Linear					
LoRA (Hu et al., 2021)	Learning rate	2e-4					
	Optimizer	AdamW					
	LR Schedule	Linear					
	rank (r)	16 (all-linear)					
	scaling (α)	8					
Flora (Hao et al., 2024b)	Learning rate	5e-4					
	LR Schedule	Constant					
	rank (r)	16 (all-linear)					
	scaling (α)	8					
	κ	1000					
LST (Sung et al., 2022)	Learning rate	3e-4					
	Optimizer	AdamW					
	LR Schedule	Linear					
	reduction factor	8					
	Init. of side network	Random					
PoWER (Goyal et al., 2020)	Learning rate	2e-4					
	Optimizer	AdamW					
	LR Schedule	Linear					
	Max seq. ratio	0.9					
	Min seq. ratio	0.3					
VeLoRA (Miles et al., 2024)	Learning rate	2e-4					
	Optimizer	AdamW					
	LR Schedule	Linear					
	rank (r)	1					
	num groups	32					
AdaZeta (Yang et al., 2024a)	Learning rate	1e-4					
	LR Schedule	Constant					
	ϵ	1e-3					
	Sparse ratio	0.85					