Variable-Rank Adaptation: A Comprehensive Survey of Rank Budgeting LoRA Variants

Anonymous ACL submission

Abstract

Low-rank adaptations (LoRA) fix the rank uniformly across all layers. Variable-rank extensions relax this constraint by assigning distinct ranks to different layers or modules, yet the resulting landscape becomes fragmented and lacks a systematic conceptual framework. We propose a unified taxonomy grouping these approaches into four categories: Rank growth, pruning, redistribution, and pre-allocation. We analyze each in terms of its design space, implementation complexity, theoretical expressiveness, and memory efficiency. Further, we identify open challenges that inhibit broader adoption: the need for mature reference implementations, comprehensive evaluation, and rigorous metrics for training and inference memory requirements.

1 Introduction

Low-rank adaptation (LoRA, Hu et al., 2021) efficiently adapts large models enabling efficient multi-tenant setups (Chen et al., 2023; Haller et al., 2024; Sheng et al., 2024) by reparametrizing the weight update $\Delta W \in \mathbb{R}^{d_2 \times d_1}$ into two smaller matrices (we omit the constant scaling factor for simplicity):

$$\Delta W = BA$$

with $B \in \mathbb{R}^{d_2 \times r}$ and $A \in \mathbb{R}^{r \times d_1}$. Using a small rank r, the required memory also remains small.

While vanilla LoRA may target only certain linear layers: e.g., only feedforward layers (esp. for knowledge-intensive tasks; Geva et al., 2021) or only certain attention projections (John Schulman, 2025; Hu et al., 2021), it applies a *constant rank* r to each targeted layer (Figure 1, top). However, layers vary in task importance (Lee et al., 2019; Houlsby et al., 2019; Vanderreydt et al., 2023; Kaplun et al., 2023; Garbaciauskas et al., 2024; Pan et al., 2024; Ke et al., 2024; Gu et al., 2024a; Gao et al., 2025; Skean et al., 2025), making uniform layouts seem inefficient. The *variable-rank* alternative assigns module-specific ranks (Fig. 1, bottom), promising better expressivity under fixed parameter budgets (Zhang et al., 2023b; He et al., 2026).

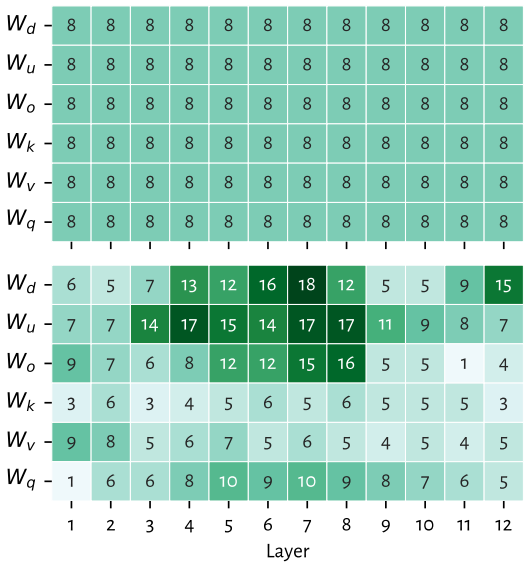
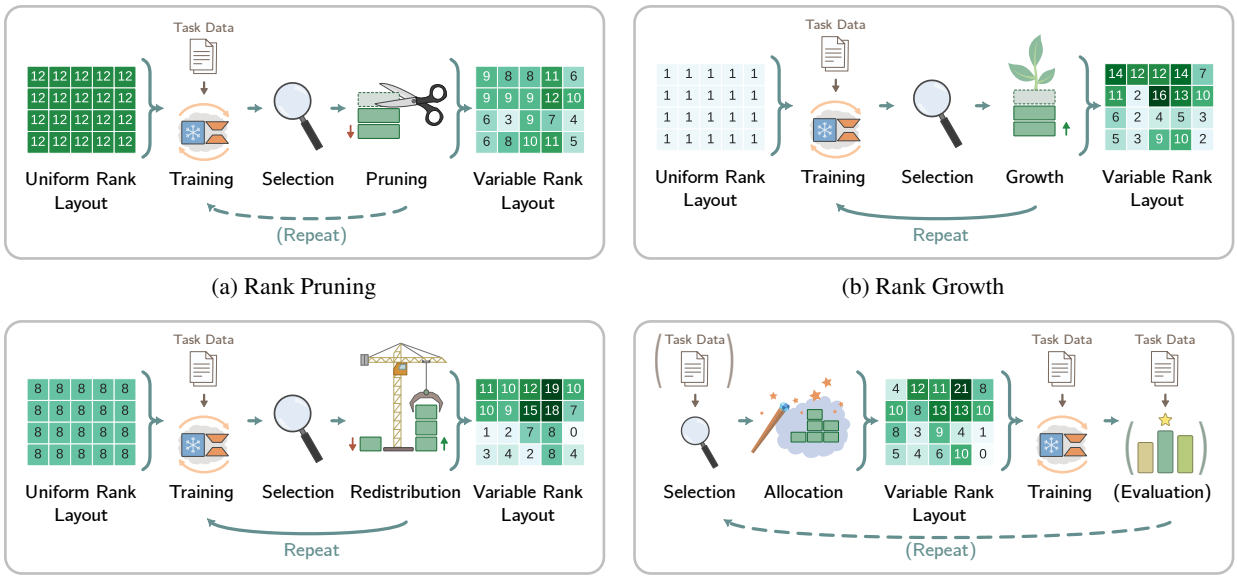


Figure 1: **Uniform**- (top) vs. **variable**-rank (bottom) configurations used for low-rank adaptation of a transformer model.

The natural question then arises: How to determine the optimal rank-layout for a given model and task? Several methods approach this question from multiple angles, often facing shared design questions: Which ranks to prune/grow? When should (growth/pruning) interventions take place? And should ranks be pushed towards orthogonality?

Yet no coherent framework or systematic discussion of these design dimensions exists, and cross-comparisons are sparse: Of the 55 papers in this survey, only six are cited by more than five other papers in the set, and only ten cite more than three other papers in the set. Where citations do occur, they often lack empirical evaluations (related work is mentioned without testing against it on the same task). This prevents conclusive comparisons between methods both within and between taxonomic groups.



(a) Rank Pruning

(b) Rank Growth

(c) Rank Redistribution

(d) Rank Pre-Allocation

Figure 2: Illustration of the four method groups.

Contribution. In this survey, we introduce a taxonomy grouping 57 variable-rank methods (across 55 publications) into four subclasses (see Figure 2) and characterize choices made during their design:

1. **Rank pruning** initializes all modules with a uniform rank (e.g. $1.5\times$ the planned average rank), then prunes individual ranks over the course of training. This yields a variable-rank configuration at the end of training.
2. **Rank growth** initializes modules uniformly with a very low rank (e.g., 0 or 1). During training, individual modules are identified for which the rank is increased.
3. **Rank redistribution** starts with a uniform rank layout that matches the planned average. During training, modules are removed and distributed among remaining modules or assigned to selected modules (as in rank growth). The average rank remains fixed during training.
4. **Rank pre-allocation** determines module-specific ranks prior to training, based either solely on the model or using model activations/gradients. The model is then trained as in vanilla LoRA, but with a variable-rank layout. *Iterative pre-allocation* extends this by training multiple models and incorporating the evaluation performance of previous layouts to determine the next.

For each category, we analyze the design dimensions, implementation complexity, theoretical expressiveness, and memory efficiency.

2 Related Work

Parameter-efficient fine-tuning methods (PEFT; Houlsby et al., 2019) inject (or select) a small set of trainable parameters while keeping the bulk of the pretrained model frozen. This dramatically reduces both computational cost and memory footprint, enabling practitioners to adapt state-of-the-art models with limited resources.

To our knowledge, there are no surveys that specifically cover (and categorize) variable-rank methods. However, multiple surveys map PEFT and LoRA approaches in differing granularity. We discuss these by increasing granularity. Our taxonomy can then be embedded into existing ones as illustrated in Figure 3.

2.1 PEFT-centric Taxonomies

Generally, there appears to be consensus over three subclasses of PEFT methods (Lialin et al., 2023; Xu et al., 2023; Han et al., 2024; Wang et al., 2025b): (1) *Additive* methods add new parameters to the existing base model (this includes adapters and soft prompts), (2) *selective* (or partial) methods select a subset of the base model parameters to tune, and (3) *reparametrization-based* fine-tuning methods produce an efficient parameterization of a parameter difference: $W_{tuned} = W_{base} + \Delta W$ with ΔW compactly parameterized.

Since there can be substantial overlap between these subclasses (Lialin et al., 2023 hence visualize it as a Venn diagram), some additionally include a "hybrid" group (Xu et al., 2023; Han et al., 2024;

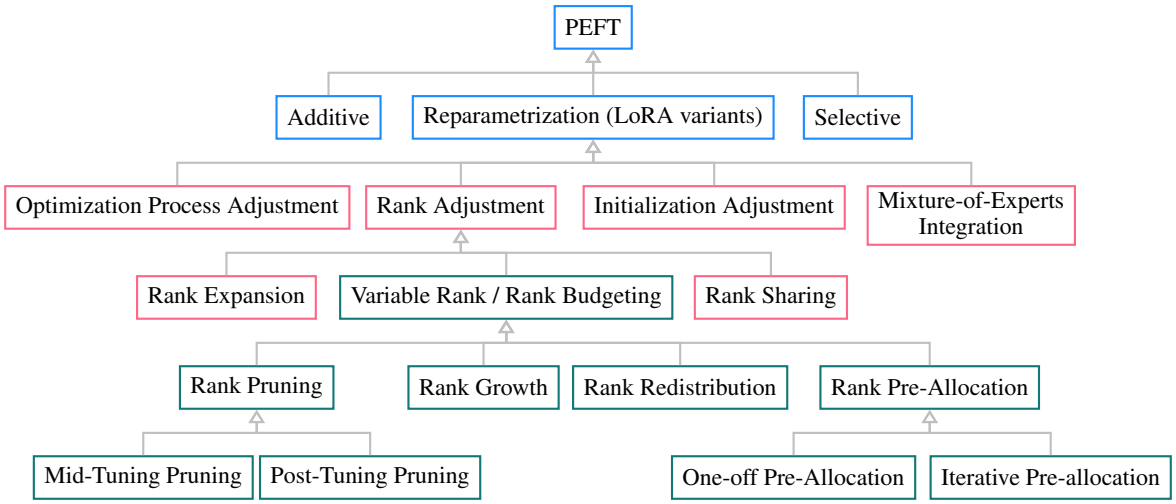


Figure 3: Taxonomy (our addition in green) based on Lialin et al. (2023, in blue) and He et al. (2026, in red).

Wang et al., 2025b). Hybrid variants between two (or more) subclasses are possible making the subclasses non-exclusive. However, hybrid variants could be categorized in two or more subclasses (not requiring an additional hybrid subclass) and we exclude the hybrid subclass from our taxonomy visualized in Figure 3.

Fine-tuning methods specifically using some form of quantization (e.g. QLoRA, Dettmers et al., 2023) may constitute another additional method group (Wang et al., 2025b). While quantization does lower memory requirements (it is a memory-efficient method), it does not (strictly speaking) in itself reduce the *number of parameters*.

2.2 LoRA-centric Taxonomies

Xu et al. (2023) and Wang et al. (2025b) split the reparameterization-based PEFT approaches into "LoRA Derivatives" and "Low-rank Decomposition", the later of which includes LoRA. With our taxonomy, we extend the work of He et al. (2026), who group methods based on functional aspects (see Figure 3). They further subdivide the field of Low-Rank Adaptation into 4 subgroups: (1) *Optimization process adjustment* (aiming to enhance stability and improve alignment with full fine-tuning), (2) *initialization adjustment* (changing how the initial rank parameters are set), (3) *mixture-of–experts integration-based* (enhancing the capacity of standard LoRA layers), and (4) *Rank adjustment* (see below),

Rank Adjustment methods aim to overcome the fixed low-rank constraint of LoRA. They are further split into three subgroups: (1) *Rank expansion*[1] increases the effective rank of a given adapter by combining multiple matrices (e.g., by using the Hadamard product[2]). (2) *Rank sharing* methods share parameters across multiple modules, reducing the overall parameter count while possibly maintaining a high rank per module. (3) *Rank budgeting* methods allocate different ranks to different modules (i.e., variable ranks).

Rank Budgeting, a subclass of rank adjustment, resembles our concept of variable-rank methods, but requires methods to impose a predefined rank budget. Most methods we survey adhere to this definition. Some, however, do not use a predefined rank budget but instead use other thresholds to determine the number of ranks. Regardless, they are rather similar to the other methods in this group and also produce variable-rank layouts. Hence, we replace 'rank budgeting' with 'variable-rank' (slightly expanding the scope and placing it in the same location in this taxonomy).

For a comparison of the variable-rank group to similar groups in other taxonomies, see Section D in the appendix.

3 Variable-Rank Methods

In this section, we introduce the four method groups and discuss the ♦ design dimensions involved in each.

[1] 'Rank expansion' here does *not* refer to our rank growth subclass, as the expansion is not applied during training but as a product of the employed mathematical operations.

[2] The Hadamard of two matrices has an effective rank equal to the product of the two ranks: when using two matrices of rank 8, the effective rank becomes 64, while addition would only yield a rank of 16.

3.1 Group 1: Rank Pruning Approaches

Rank pruning approaches start with a uniform rank layout with a high initial rank, select the least important ranks, and remove those (see Figure 2). Removing more ranks in some modules than others leaves a non-uniform rank layout with a lower average rank than the initial layout.

Pruning methods generally exhibit four design dimensions:

- ♦ **selection criterion:** Which ranks to remove.
- ♦ **parametrization:** How the weight update ΔW is decomposed (often intertwined with the ♦ selection criterion).
- ♦ **orthonormalization:** How to keep the rank vectors at unit length and orthogonal, ensuring coefficients accurately reflect each rank's contribution.
- ♦ **sparsity:** How to enforce sparsity, whether by removing ranks already during training or through other means.

We distinguish between two subgroups of rank pruning approaches: *Mid-tuning* in which ranks are progressively pruned during training, and *post-tuning*, where ranks are pruned only once after training is complete. In the following subsections, we give details on both subgroups.

3.1.1 Mid-Tuning Rank Pruning

The first subgroup contains approaches that remove ranks during training. In our survey, we identified 15 distinct approaches within this subgroup (see Table 1 in the appendix for a full list).

The main motivation for these approaches is the tendency of pruning to disrupt the model's function, as other components were trained with those elements in place. Therefore, these mid-tuning approaches prune already during training employing a single or multiple small pruning steps, imposing ♦ sparsity by design, and giving the model the chance to adapt to the removal of ranks.

Design Dimensions. The 15 approaches surveyed differ along the following design dimensions:

♦ **selection criterion.** One of the most important decisions is to determine which ranks to remove. Different approaches use different metrics such as parameter-magnitude (of the base weight or the learned rank vectors; e.g., Kim et al., 2024a; Sharma et al., 2025), gradient- and activation-information (e.g., Lawton et al., 2023; Zhou et al., 2025b), or some learned gating parameters (Hu et al., 2023). Parameter-magnitude, learned gating parameters, and activation patterns can indicate each rank's influences on the (next) latent representation, while gradient information additionally accounts for the impact on the loss.

♦ **parametrization.** Instead of using LoRA's parametrization of $\Delta W = BA$, six of the 15 mid-tuning approaches use a diagonal matrix Λ to regulate the contribution of each rank (resembling the result of a singular value decomposition):

$$\Delta W = P\Lambda Q$$

This can have two benefits for variable-rank LoRA approaches: First, the contribution of a rank can easily be adjusted by setting the corresponding value in Λ (e.g., to zero to disable a rank completely). This contribution can either be learned directly or computed based on some other function. Second, information on Λ (and its gradient) can be used in the ♦ selection criterion since it represents the ranks' contribution – ranks with lower contribution may be less relevant to the models function.

♦ **orthonormalization.** Values in Λ in SVD-like ♦ parametrizations only represent the contribution, if rank vectors have unit length. Further, recent work has shown empirical benefits of orthogonal ranks for uniform rank adaptation (Li et al., 2025) and theoretical analysis suggests that gradient-based optimization of factorized adapters may yield suboptimal results (Schotthöfer et al., 2025).

Unit length can be enforced via regularization or by normalization after each training step. Similarly, orthogonality can be achieved either by adding a penalty to the loss or by projecting the current vectors onto orthogonal vectors, e.g., using the QR algorithm (Francis, 1961; Schotthöfer et al., 2025) or the Gram-Schmidt process (Gander, 1980).

Some methods use loss-based regularization to make vectors orthogonal and of unit length, while most do not apply any orthonormalization at all. Prune LoRA (a baseline introduced together with AdaLoRA; Zhang et al., 2023b) and SalientLoRA (Ke et al., 2024) use the following orthogonal regularization (with $\|\cdot\|_F$ being the Frobenius norm):

$$R_{orth}(B, A) = \|B^\intercal B - I\|_F^2 + \|AA^\intercal - I\|_F^2$$

This penalizes pairs of vectors that are not orthogonal (with a non-negligible inner product) and vectors deviating from a norm of one.

3.1.2 Post-Tuning Rank Pruning

The second subgroup consists of approaches that instead do a single pruning step after training is complete. As discussed, one issue is that the sudden removal of so many ranks may incapacitate the model. To address this, *post-tuning* needs to minimize the impact of pruning during training; typically by promoting ♦ sparsity over the course of the training through other means. Often, an SVD-like ♦ parametrization that includes a rank coefficient/gating value is used.

Design Dimensions. In our survey, we identified 15 different approaches (see Table 2 in the appendix for a full list) that differ along the same three dimensions as mid-tuning approaches (♦ selection criterion, ♦ parametrization, ♦ orthonormalization), but add the crucial ♦ sparsity dimension that determines how rank sparsity is promoted.

♦ sparsity. Three primary strategies promote sparsity within the learned ranks.

Iterative coefficient zeroing progressively sets larger subsets of the rank coefficients to zero based on the ♦ selection criterion (first in AdaLoRA; Zhang et al., 2023b). Unlike mid-tuning, which removes rows in A (or columns in B), permanently losing the directional information, iterative zeroing allows recovery by increasing the coefficient again.

The shrinkage operator zeroes values with magnitudes below a threshold ξ, and shrinking others by the same value. This soft-thresholding approach is commonly used in proximal gradient methods for ℓ_1 penalties (Beck and Teboulle, 2009). Examples include SoRA (Ding et al., 2023), CoDyRA (Lu et al., 2026), and RaLo (Deng et al., 2026b).

Stochastic selection deactivates ranks during training forcing the model to adapt to varying sets of active ranks. For instance, LoNAS (Munoz et al., 2024) deactivates modules below a sampled boundary layer, B-LoRA (Meo et al., 2024) drops out ranks based on learned rank-specific dropout probabilities, and IST (Yao et al., 2024) samples modules via importance scores.

Inherent Properties & Limitations

Rank pruning approaches have a strict upper limit on the rank number for each module, given by the initial rank that the adapters are trained with. Since memory, in practical settings, is limited, the initial rank may be limited as well. Depending on the scenario, this can greatly limit rank variability.

While post-tuning approaches maintain a fixed total rank over the course of training, mid-tuning approaches reduce the total rank even during training. However, in practical approaches, this reduction in memory is not utilized (for a detailed discussion, see Section E in the appendix).

3.2 Group 2: Rank Growth Approaches

Like rank pruning, rank growth approaches change the rank number of modules over time. Instead of reducing it, they increase it. One advantage is that, unlike rank pruning approaches, there is no pre-determined maximum rank a module may receive. Since growth happens over the course of training (where module are progressively identified that are increased in rank), they share many design dimensions of mid-tuning rank pruning: ♦ selection criterion to select which ranks to grow, ♦ parametrization, and ♦ orthonormalization.

One important additional design dimension is ♦ initialization, which determines how a newly added rank is initialized so as to best increase model capacity. A poorly initialized new rank might otherwise simply be ignored during training if the rest of the model is already well aligned.

Design Dimensions. In our survey, we identified 7 distinct rank growth approaches (see Table 3 in the appendix). The two most important design dimensions are ♦ initialization and ♦ selection criterion:

♦ selection criterion. Rank growth approaches differ in what kind of signals they exploit: gradient-based sensitivity (IncreLoRA; Zhang et al., 2023a), with statistics computed during backpropagation to avoid storing all gradients), reconstruction error from base weight pruning (LoSA; Huang et al., 2025), and activation convergence (AROMA; Sheng et al., 2025, which iteratively adds ranks until the change in output activation norm falls below a threshold). Others leverage routing frequency (DR-LoRA; Deng et al., 2026a) or empirical Fisher Information ratios between current and expanded ranks (SeLoRA; Mao et al., 2026).

♦ initialization. Most methods initialize new ranks similar to the initial ranks, preserving the function (see also Chen et al., 2016) by setting the respective elements in the B for LoRA-like parametrization or the singular values in a SVD-like parametrization to zero. New parameters also require a learning rate schedule. The method must determine whether to use the same (post-warmup) learning rate or a dedicated schedule (potentially with warmup and decay). Since none of the papers

discuss this choice, we assume new parameters follow the current global learning rate.

Flexible Rank. GeoLoRA (Schotthöfer et al., 2025) is an outlier in this group as it can also decrease a module's rank (or even the overall total). We do not classify it as a redistribution method since the total rank (or parameter) budget is not maintained. Instead, ranks are doubled in each augmentation step, and those below a certain singular value threshold are removed again. This enables the approach to grow *and* shrink.

Inherent Properties & Limitations

Rank growth methods are typically less constrained in the maximum final rank. However, DR-LoRA (Deng et al., 2026a) and La-LoRA (Gu et al., 2026) pre-allocate a fixed rank budget, upper-bounding any module's final rank. Like mid-tuning rank pruning approaches, rank growth methods exhibit a changing total rank throughout training. Since memory usage peaks at the end of training, this should inform choices of memory-related hyperparameters such as sequence length and batch size.

3.3 Group 3: Rank Redistribution Approaches

Rank redistribution methods share aspects of the mid-tuning rank pruning group as well as the rank growth groups: They prune certain ranks and extend the ranks in other modules, maintaining a constant total/average rank. They include all design dimensions of these two groups (which we do not discuss in detail here again as the set of choices is quite similar; see Table 4 for the five approaches), but add a crucial new dimension:

- ♦ **redistribution mode:** A unique feature of rank redistribution methods is how they select ranks/-modules and redistribute de-allocated ranks.

Most methods follow a bottom-k pruning, top-k expansion scheme: ALoRA and L1RA use round-robin expansion of non-pruned modules (Liu et al., 2024; Singh et al., 2025), while ElaLoRA expands top-k modules by minimal importance with orthogonal Gram-Schmidt initialization (Chang et al., 2025). WeightLoRA+ similarly expands non-pruned modules but prunes at the module level (Veprikov et al., 2025), and FlexLoRA performs coupled rank removal and addition across bottom-k and top-k modules respectively (Liu et al., 2026).

Inherent Properties & Limitations

Unlike rank pruning and growth approaches, rank redistribution approaches maintain a constant total rank. While total memory use may still vary (some ranks require more parameters than others), this variation is typically much smaller. Just like rank growth methods, rank redistribution methods are not constrained by an initial rank. Instead, selected modules can, in principle, grow quite large.

3.4 Group 4: Rank Pre-Allocation Approaches

While the previous groups exhibit dynamic ranks across training (i.e., they change the rank number for a given module during training), this is not required to produce variable-rank layouts: Rank pre-allocation determines the rank numbers once prior to training, and then continues to train the adapters without making any more structural changes.

3.4.1 One-off Pre-Allocation

The simplest variant is to produce a variable-rank layout prior to training and train only once. For detailed choices of how this is achieved, see Table 5 in the appendix.

- ♦ **information source:** Pre-allocation methods determine rank layouts prior to training using diverse ♦ criteria, based on different types of information, such as base weights, task-data activations, gradients, or prior experimental results.

Most of the 11 approaches require the target task data to be passed through the model (see Table 5) to collect hidden states (activations), gradient information, or even the diagonal of the Hessian matrix (i.e., second-order derivatives).[3] PRILoRA (Benedek and Wolf, 2024) uses a static rank layout (determined based on previous experiments, independent of the base model and target task). SARA (Gu et al., 2024a) and SR-LoRA (Zhang et al., 2025a) create the rank layout based on the base weights. Similarly, RA-LoRA (Kim et al., 2024b) bases the decision on the quantization error.

♦ **initialization.** Once the rank layout has been determined, some methods initialize rank parameters like LoRA. GoRA (He et al., 2025) initializes B such that BA approximates the gradient of the

[3] Where gradients are required, they need to be computed for the whole weight matrices, potentially across multiple minibatches. Since the base weights are not optimized, only the model parameters, as well as the gradient, but no additional statistical moments of the gradient need to be kept (as required by some optimizers; Kingma and Ba, 2015).

base weight (using the scaled Moore-Penrose inverse of A), RaLoRA-(Pro) initializes like MeLoRA (Ren et al., 2024) which it is based on, and EVA (Paischer et al., 2025) sets A based on the principal directions of the input activations to speed up the initial training by maximizing the gradient of B, which is initialized as zero (like in LoRA).

3.4.2 Iterative Pre-Allocation

Iterative pre-allocation repeatedly trains models with new rank layouts and uses performance metrics to influence the selection of new layouts. We specifically exclude trial-based search approaches, which only search for a global (uniform) rank r, but do include approaches that train multiple non-uniform layouts (see Table 6 for details). We include four approaches in the group.

- ♦ **search algorithm:** Candidate layouts are proposed using different algorithms. Information on the performance on validation data is then used to refine the search process

Some methods use this information to train a performance model (as in the case of RankAdaptor and HyDRA, Zhou et al., 2025a; Xi et al., 2025) or a layout-generating hyper-network (LARA, Zhou et al., 2026b), or to inform the evolutionary search process (AutoQRA, Zhou et al., 2026a).

Inherent Properties & Limitations

The properties and limitations differ drastically between one-off and iterative approaches. Generally, the training process for pre-allocation methods can be implemented as efficiently as uniform-rank approaches. Once the layout is determined, the model is not modified during training, and memory use only depends on the processed data.

One-off pre-allocation cannot adapt the layout once training starts, making it relatively inflexible. However, existing LoRA-training tooling can be used: no additional metrics, interventions, or hyperparameters are needed. We hypothesize that while complex tasks may require more flexibility during fine-tuning, for most tasks, the simplicity of one-off pre-allocation outweighs the inflexibility.

Iterative pre-allocation differs substantially from the other groups, as it trains multiple models that are then evaluated on held-out validation data. While an empirical comparison is lacking, iterative pre-allocation likely requires significantly more computing resources. It is unclear how much of this can be offset by sharing performance predictors or rank layout-generating hyper-networks across multiple tasks and base models.

3.5 Additional Design Dimensions

Two additional design dimensions apply to all variable-rank approaches.

- ♦ **granularity:** While rank-level approaches assign a rank of $r_m \in \mathbb{N}_0$ to each module m, some approaches select modules (e.g., key projection in the 7th layer) or even complete layers where LoRA adapters are applied (i.e, for each module they set a rank of $r_m \in \{0, r_{shared}\}$).
- ♦ **base model compression:** Dettmers et al. (2023) showed that quantization of the base model enables fine-tuning of much larger models with limited memory. While quantization of the base model may seem orthogonal to the employed fine-tuning technique, some works propose joining base model compression with task-specific variable-rank fine-tuning.

RA-LoRA (Kim et al., 2024b) and LoSA (Huang et al., 2025) use the reconstruction error to inform the rank layout. LoSA, RankAdaptor (Zhou et al., 2025a), LARA (Zhou et al., 2026b), and Light-PEFT (Gu et al., 2024b) use parameter pruning to compress the base model; RA-LoRA uses quantization. Since only these five approaches employ base model compression, we omit this design dimension from Tables 1–6.

4 Open Challenges and Opportunities

In this section, we discuss some practical and theoretical challenges and opportunities.

4.1 Practical Evaluation Obstacles

A major limitation is the lack of comprehensive evaluations. Current publications use diverse settings ranging from simple natural language understanding tasks (such as classification tasks, e.g., on GLUE data; Wang et al., 2018) to natural language generation (multiple choice questions, or long-form question answering; e.g. GSM8K; Cobbe et al., 2021). Large-scale evaluations are lacking, particularly for compute and memory efficiency. This limitation broadly extends to PEFT, where the standardization of setups and development of metrics measuring training and inference costs remain ongoing (Belanec et al., 2026).

Replication studies are labor-intensive to conduct, especially since source code is easily retriev-

able only for approximately half of the papers.[4] Even when source code is provided, it (1) follows diverse package structures, (2) relies on various libraries, often with copied dependencies, making modifications difficult to locate, and (3) sometimes even requires adjustments for different models. Many of the approaches not only require the injection of specific adapters into the model, but also the collection of activation and gradient statistics, as well as changes to the model during training or, in some cases, specialized schedulers. Without fully documented training scripts, concrete implementation details may be unclear.

Further, hyperparameter robustness remains underexplored. Especially for approaches with multiple interacting components (e.g., learning of existing ranks as well as selecting the right ranks to add/remove), the performance gained by introducing additional brittleness needs careful study.

These obstacles hinder both practical adoption and further scientific inquiry. He et al. (2026) evaluated 20 LoRA variants, only 2 of which can be categorized as rank budgeting methods (AdaLoRA and EVA, the second of which was also classified as an initialization adjustment-based approach). Without established baselines, future approaches must re-implement baselines for adequate comparison.

4.2 Higher Rank Does Not Always Lead to Higher Performance

While an insufficiently large rank can produce subpar results (especially in knowledge-intensive fine-tuning tasks Back et al., 2026), increasingly high ranks may yield diminishing returns or even reduce performance (Zhang et al., 2023b; Dettmers et al., 2023). Wang et al. (2023) find that with small training datasets high-rank LoRA modules may be overparameterized. This contrasts with most variable-rank methods, which assume all ranks improve performance.

Further, most approaches require the developer to set a rank budget. A truly automatic rank selection method should reduce the average rank if increasing the ranks may cause overfitting. A few methods use threshold values to determine how many ranks are added/removed (Ding et al., 2023; Gu et al., 2024a; Schotthöfer et al., 2025; Mao et al., 2026). However, these thresholds are often difficult to interpret, replacing one hyperparameter with another. Ideally, the developer could easily determine what performance gain justifies a rank increase, while the method handles the rest and avoids excessive ranks.

4.3 Not Every Rank has Equal Size

Not every rank induces the same computational and memory cost, so comparing rank-configurations solely by average rank may be insufficient. For example, a single rank in a linear layer of a feed-forward block of SmolLM2 or PaLM requires 2.5 times the parameters of one in a self-attention block (Allal et al., 2025; Chowdhery et al., 2023).

While in PEFT, it is good practice to plot performance over the number of adapted parameters; few papers (Gao et al., 2024; Ye et al., 2025; He et al., 2025) actively consider the cost of adding a rank over another with a different parameter count. This may leave some potential for improvement, as some performance could be gained by replacing a certain rank with multiple others matching the parameter count. Furthermore, information about the parameter count is often not included when representing the results in other forms than the parameter count vs. performance plot.

5 Conclusion

This paper systematically analyzes methods producing variable-rank layouts for low-rank fine-tuning categorizing 57 methods from 55 papers into four main groups: rank pruning, rank growth, rank redistribution, and rank pre-allocation. By mapping the current landscape, clarifying key gaps, and proposing directions for future work, we hope to equip future research towards more practical and scalable variable-rank methods.

Despite the progress in this field, practical and methodological challenges hinder broader adoption: notably, mature reference implementations, comprehensive evaluations, and rigorous metrics to accurately assess memory requirements during both training and inference phases remain lacking.

Most works focus on single-task transfer learning, yet variable-rank methods may be particularly suited for continual learning settings, where some ranks may be adapted to work for multiple tasks, while others may serve specific tasks. While recent works (Kim et al., 2026) started exploring this direction, substantial room for improvement remains.

[4] 18 out of 55 of the papers provide a URL to a repository (excluding two where it led to a non-existing page or did not contain any code). In 9 cases, we could find repositories through manual search. In one case, we found a third-party implementation where no author-provided repository was found.

Limitations

Though we did not explicitly exclude non-English papers, the initial set of seed papers only contained English papers, and no non-English papers were added during the expansion phase. We cannot guarantee that the list of analysis papers is exhaustive. However, we are confident any additional paper would not be directly connected to any of the included papers via citation (i.e., be relatively disconnected; see Section A in the appendix for details).

We are not comparing methods based on reported performance, but solely based on the strategies employed in the methods as such. Hence, papers referenced for a method included in our taxonomy were not scrutinized for the scientific rigor of the experimental section. A few papers were excluded based on quality issues, regardless (see Section A).

Finally, our survey does not include an empirical comparison and reproduction of the reported papers. While such a comparison would be quite valuable, it would require (1) immense manual work in integrating and re-implementing a large number of methods, and (2) significant computational resources to train and evaluate all of the methods on a decent number of tasks.

References

Aastik, Meghana Topu, Chinmay Kulkarni, and Pragya Paramita Sahu. 2025. NormAL LoRA: What is the perfect size? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19716–19731, Suzhou, China. Association for Computational Linguistics.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model.

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. 2026. Understanding LoRA as Knowledge Memory: An Empirical Analysis.

Amir Beck and Marc Teboulle. 2009. A fast Iterative Shrinkage-Thresholding Algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696.

Robert Belanec, Branislav Pecher, Ivan Srba, and Maria Bielikova. 2026. PEFT-Bench: A Parameter-Efficient Fine-Tuning Methods Benchmark.

Nadav Benedek and Lior Wolf. 2024. PRILoRA: Pruned and Rank-Increasing Low-Rank Adaptation. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 252–263, St. Julian's, Malta. Association for Computational Linguistics.

Huandong Chang, Zicheng Ma, Mingyuan Ma, Zhenting Qi, Andrew Sabot, Hong Jiang, and H. T. Kung. 2025. ElaLoRA: Elastic & Learnable Low-Rank Adaptation for Efficient Model Fine-Tuning.

Guanhua Chen, Yutong Yao, Ci-Jun Gao, Lidia S. Chao, Feng Wan, and Derek F. Wong. 2025a. Not All LoRA Parameters Are Essential: Insights on Inference Necessity.

Haolin Chen and Philip N. Garner. 2025. A Bayesian Interpretation of Adaptive Low-Rank Adaptation. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. 2023. Punica: Multi-Tenant LoRA Serving.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2016. Net2Net: Accelerating Learning via Knowledge Transfer. *arXiv:1511.05641 [cs]*.

Zezhou Chen, Zhaoxiang Liu, Kai Wang, and Shiguo Lian. 2025b. Reparameterization-Based Parameter-Efficient Fine-Tuning Methods for Large Language Models: A Systematic Survey. In *Natural Language Processing and Chinese Computing*, pages 107–118, Singapore. Springer Nature.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi,

David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems.

Guanzhi Deng, Bo Li, Ronghao Chen, Huacan Wang, Lijie Wen, and Linqi Song. 2026a. DR-LoRA: Dynamic Rank LoRA for Mixture-of-Experts Adaptation.

Yunsong Deng, Guoxu Zhou, and Qibin Zhao. 2026b. RaLo: Rank-aware low-rank adaptation for pretrained foundation models. *Neural Networks*, 196:108423.

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.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse Low-rank Adaptation of Pre-trained Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145, Singapore. Association for Computational Linguistics.

Yihe Dong. 2024. AdaRank: Disagreement Based Module Rank Prediction for Low-rank Adaptation.

Abdessalam Ed-dib, Zhanibek Datbayev, and Amine M. Aboussalah. 2025. GeLoRA: Geometric Adaptive Ranks For Efficient LoRA Fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25174–25196, Suzhou, China. Association for Computational Linguistics.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. 2022. KronA: Parameter Efficient Tuning with Kronecker Adapter.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2017. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7(1):12140.

J. G. F. Francis. 1961. The QR Transformation A Unitary Analogue to the LR Transformation—Part 1. *The Computer Journal*, 4(3):265–271.

Walter Gander. 1980. Algorithms for the QR-Decomposition. Technical Report 80-02, Eidgenössische Technische Hochschule, Zürich.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and Vs Subrahmanian. 2025. MoLA: MoE LoRA with Layer-wise Expert Allocation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5097–5112, Albuquerque, New Mexico. Association for Computational Linguistics.

Shangqian Gao, Ting Hua, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. 2024. Adaptive Rank Selections for Low-Rank Approximation of Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 227–241, Mexico City, Mexico. Association for Computational Linguistics.

Yihang Gao and Vincent Y. F. Tan. 2026. Automatic Rank Determination for Low-Rank Adaptation via Submodular Function Maximization. *IEEE Transactions on Signal Processing*, 74:1094–1108.

Lukas Garbaciauskas, Max Ploner, and Alan Akbik. 2024. Choose Your Transformer: Improved Transferability Estimation of Transformer Models on Classification Tasks. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12752–12768, Bangkok, Thailand. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiancheng Gu, Jiabin Yuan, Jiyuan Cai, Xianfa Zhou, and Lili Fan. 2026. La-LoRA: Parameter-efficient fine-tuning with layer-wise adaptive low-rank adaptation. *Neural Networks*, 194:108095.

Jihao Gu, Shuai Chen, Zelin Wang, Yibo Zhang, and Ping Gong. 2024a. SARA: Singular-Value Based Adaptive Low-Rank Adaption.

Naibin Gu, Peng Fu, Xiyu Liu, Bowen Shen, Zheng Lin, and Weiping Wang. 2024b. Light-PEFT: Lightening Parameter-Efficient Fine-Tuning via Early Pruning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7528–7541, Bangkok, Thailand. Association for Computational Linguistics.

Patrick Haller, Ansar Aynetdinov, and Alan Akbik. 2024. OpinionGPT: Modelling Explicit Biases in Instruction-Tuned LLMs. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies (Volume 3: System Demonstrations), pages 78–86, Mexico City, Mexico. Association for Computational Linguistics.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *Transactions on Machine Learning Research*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient Low Rank Adaptation of Large Models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pages 17783–17806, Vienna, Austria. JMLR.org.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2025. PLoP: Precise LoRA Placement for Efficient Fine-tuning of Large Models.

Haonan He, Jingqi Ye, Minglei Li, Zhengbo Wang, Tao Chen, Lei Bai, and Peng Ye. 2026. A Unified Study of LoRA Variants: Taxonomy, Review, Codebase, and Empirical Evaluation.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. 2025. GoRA: Gradient-driven Adaptive Low Rank Adaptation.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzebski, and Bruna Morrone. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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.

Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. 2023. Structure-Aware Low-Rank Adaptation for Parameter-Efficient Fine-Tuning. *Mathematics*, 11(20):4317.

Weizhong Huang, Yuxin Zhang, Xiawu Zheng, Yang Liu, Jing Lin, Yiwu Yao, and Rongrong Ji. 2025. Dynamic Low-Rank Sparse Adaptation for Large Language Models.

Tangyu Jiang, Haodi Wang, and Chun Yuan. 2025. DifforA: Enabling Parameter-Efficient Fine-Tuning via Differential Module Selection.

John Schulman. 2025. LoRA Without Regret.

Damjan Kalajdzievski. 2023. A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA.

Gal Kaplun, Andrey Gurevich, Tal Swisa, Mazor David, Shai Shalev-Shwartz, and Eran Malach. 2023. Less is More: Selective Layer Finetuning with SubTuning.

Wenjun Ke, Jiahao Wang, Peng Wang, Jiajun Liu, Dong Nie, Guozheng Li, and Yining Li. 2024. Unveiling LoRA Intrinsic Ranks via Salience Analysis. In *Advances in Neural Information Processing Systems*, volume 37, pages 131575–131595.

Donghoon Kim, Minji Bae, Unghui Nam, Gyeonghun Kim, Suyun Lee, Kyuhong Shim, and Byonghyo Shim. 2026. Adaptive Capacity Allocation for Vision Language Action Fine-tuning.

Juhyeong Kim, Gyunyeop Kim, and Sangwoo Kang. 2024a. Lottery Rank-Pruning Adaptation Parameter Efficient Fine-Tuning. *Mathematics*, 12(23).

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, Bangkok, Thailand. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, Miles Crawford, Doug Downey, Jason Dunkelberger, Oren Etzioni, Rob Evans, Sergey Feldman, Joseph Gorney, David Graham, Fangzhou Hu, Regan Huff, Daniel King, Sebastian Kohlmeier, Bailey Kuehl, Michael Langan, Daniel Lin, Haokun Liu, Kyle Lo, Jaron Lochner, Kelsey MacMillan, Tyler Murray, Chris Newell, Smita Rao, Shaurya Rohatgi, Paul Sayre, Zejiang Shen, Amanpreet Singh, Luca Soldaini, Shivashankar Subramanian, Amber Tanaka, Alex D. Wade, Linda Wagner, Lucy Lu Wang, Chris Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Madeleine Van Zuylen, and Daniel S. Weld. 2025. The Semantic Scholar Open Data Platform.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. VeRA: Vector-based Random Matrix Adaptation.

Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. 2023. Neural Architecture Search for Parameter-Efficient Fine-tuning of Large Pre-trained Language Models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8506–8515, Toronto, Canada. Association for Computational Linguistics.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning.

Zhizhong Li, Sina Sajadmanesh, Jingtao Li, and Lingjuan Lyu. 2025. StelLA: Subspace Learning in Low-rank Adaptation using Stiefel Manifold.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning.

Muqing Liu, Chongjie Si, and Yuheng Jia. 2026. FlexLoRA: Entropy-Guided Flexible Low-Rank Adaptation.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. 2024. ALoRA: Allocating Low-Rank Adaptation for Fine-tuning Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 622–641, Mexico City, Mexico. Association for Computational Linguistics.

Haodong Lu, Chongyang Zhao, Jason Xue, Lina Yao, Kristen Moore, and Dong Gong. 2026. Adaptive Rank, Reduced Forgetting: Continual Learning with Dynamic Rank-Selective LoRA.

Yuchen Mao, Hongwei Bran Li, Wei Pang, Giorgos Papanastasiou, Guang Yang, Ehsan Mohammadipasand, Theodore Harrison-Drummond, and Chengjia Wang. 2026. *SeLoRA*: Self-expanding LoRA for high-quality and efficient medical image synthesis. *Expert Systems with Applications*, 314:131569.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. DoRA: Enhancing Parameter-Efficient Fine-Tuning with Dynamic Rank Distribution.

Cristian Meo, Ksenia Sycheva, Anirudh Goyal, and Justin Dauwels. 2024. Bayesian-LoRA: LoRA based Parameter Efficient Fine-Tuning using Optimal Quantization levels and Rank Values trough Differentiable Bayesian Gates.

Juan Pablo Munoz, Jinjie Yuan, Yi Zheng, and Nilesh Jain. 2024. LoNAS: Elastic Low-Rank Adapters for Efficient Large Language Models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10760–10776, Torino, Italia. ELRA and ICCL.

Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Peter Deisenroth, and Sepp Hochreiter. 2025. Parameter Efficient Fine-tuning via Explained Variance Adaptation.

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.

Rushi Qiang, Ruiyi Zhang, and Pengtao Xie. 2024. BiLoRA: A Bi-level Optimization Framework for Overfitting-Resilient Low-Rank Adaptation of Large Pre-trained Models.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten De Rijke, Zhumin Chen, and Jiahuan Pei. 2024. MELoRA: Mini-Ensemble Low-Rank Adapters for Parameter-Efficient Fine-Tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3064, Bangkok, Thailand. Association for Computational Linguistics.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2024. Tied-LoRA: Enhancing parameter efficiency of LoRA with Weight Tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8694–8705, Mexico City, Mexico. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Steffen Schotthöfer, Emanuele Zangrando, Gianluca Ceruti, Francesco Tudisco, and Jonas Kusch. 2025. GeoLoRA: Geometric integration for parameter efficient fine-tuning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Manish Sharma, Jamison Heard, Eli Saber, and Panagiotis Markopoulos. 2025. Convolutional Neural Network Compression via Dynamic Parameter Rank Pruning. *IEEE Access*, 13:18441–18456.

Hao Nan Sheng, Zhi-yong Wang, Mingrui Yang, and Hing Cheung So. 2025. AROMA: Autonomous Rank-one Matrix Adaptation.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. 2024. S-LoRA: Serving Thousands of Concurrent LoRA Adapters.

Haseeb Ullah Khan Shinwari and Muhammad Usama. 2025. ARD-LoRA: Dynamic Rank Allocation for Parameter-Efficient Fine-Tuning of Foundation Models with Heterogeneous Adaptation Needs. *IEEE Transactions on Artificial Intelligence*, pages 1–10.

Raul Singh, Nicolo Brunello, Vincenzo Scotti, and Mark James Carman. 2025. L1RA: Dynamic Rank Assignment in LoRA Fine-Tuning.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by Layer: Uncovering Hidden Representations in Language Models.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. 2023. Parameter-Efficient Multi-Task Model Fusion with Partial Linearization. In *The Twelfth International Conference on Learning Representations*.

Aaron Xuxiang Tian, Yi Zhao, Congrui Yin, Wei Zhu, Xing Tian, and Yi Ge. 2024. FanLoRA: Fantastic LoRAs and Where to Find Them in Large Language Model Fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 515–528, Miami, Florida, US. Association for Computational Linguistics.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. DyLoRA: Parameter-Efficient Tuning of Pretrained Models using Dynamic Search-Free Low Rank Adaptation.

Geoffroy Vanderreydt, Amrutha Prasad, Driss Khalil, Srikanth Madikeri, Kris Demuynck, and Petr Motlicek. 2023. Parameter-Efficient Tuning with Adaptive Bottlenecks for Automatic Speech Recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–7, Taipei, Taiwan. IEEE.

Andrey Veprikov, Vladimir Solodkin, Alexander Zyl, Andrey Savchenko, and Aleksandr Beznosikov. 2025. WeightLoRA: Keep Only Necessary Adapters.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Chengyu Wang, Junbing Yan, Wei Zhang, and Jun Huang. 2023. Towards Better Parameter-Efficient Fine-Tuning for Large Language Models: A Position Paper.

Huiyi Wang, Lina Yao, and Dong Gong. 2025a. Pi-LoRA: Gradient-Informed Parameter-Importance-Aware Low-Rank Adaptation.

Luping Wang, Sheng Chen, Linnan Jiang, Shu Pan, Runze Cai, Sen Yang, and Fei Yang. 2025b. Parameter-efficient fine-tuning in large language models: A survey of methodologies. *Artificial Intelligence Review*, 58(8):227.

Chenxing Wei, Yao Shu, Ying Tiffany He, and Fei Yu. 2025. Flexora: Flexible Low-Rank Adaptation for Large Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14643–14682, Vienna, Austria. Association for Computational Linguistics.

Fei Wu, Jia Hu, Geyong Min, and Shiqiang Wang. 2026. Adaptive Rank Allocation for Federated Parameter-Efficient Fine-Tuning of Language Models. *IEEE Transactions on Computers*, pages 1–14.

Jiarun Wu and Qingliang Chen. 2022. Pruning Adapters with Lottery Ticket. *Algorithms*, 15(2).

Yuanhao Xi, Xiaohuan Bing, and Ramin Yahyapour. 2025. HyDRA: Hierarchical and Dynamic Rank Adaptation for Mobile Vision Language Model. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment.

Menglin Yang, Jialin Chen, Jinkai Tao, Yifei Zhang, Jiahong Liu, Jiasheng Zhang, Qiyao Ma, Harshit Verma, Regina Zhang, Min Zhou, Irwin King, and Rex Ying. 2025. Low-Rank Adaptation for Foundation Models: A Comprehensive Review.

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.

Jingqi Ye, Haonan He, Minglei Li, Fujun Han, Tao Chen, and Peng Ye. 2025. Gradient Intrinsic Dimensionality Alignment: Narrowing The Gap Between Low-Rank Adaptation and Full Fine-Tuning. In *The Fourteenth International Conference on Learning Representations*, Rio de Janeiro, Brazil.

Xin Yu, Cong Xie, Ziyu Zhao, Tiantian Fan, Lingzhou Xue, and Zhi Zhang. 2025. PrunedLoRA: Robust Gradient-Based structured pruning for Low-rank Adaptation in Fine-tuning.

Chuyan Zhang, Kefan Wang, and Yun Gu. 2025a. Beyond Low-Rank Tuning: Model Prior-Guided Rank Allocation for Effective Transfer in Low-Data and Large-Gap Regimes.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. 2023a. IncreLoRA: Incremental Parameter Allocation Method for Parameter-Efficient Fine-tuning.

Hao Zhang, Bo Huang, Zhenjia Li, Xi Xiao, Hui Yi Leong, Zumeng Zhang, Xinwei Long, Tianyang Wang, and Hao Xu. 2025b. Sensitivity-LoRA: Low-Load Sensitivity-Based Fine-Tuning for Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13185–13199, Suzhou, China. Association for Computational Linguistics.

Haojie Zhang. 2025. DropLoRA: Sparse Low-Rank Adaptation for Parameter-Efficient Fine-Tuning.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. AutoLoRA: Automatically Tuning Matrix Ranks in Low-Rank Adaptation Based on Meta Learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 5048–5060, Mexico City, Mexico. Association for Computational Linguistics.

Yi Zhao, Qinghua Yao, Xinyuan song, and Wei Zhu. 2026. ShapLoRA: Allocation of Low-rank Adaption on Large Language Models via Shapley Value Inspired Importance Estimation.

Changhai Zhou, Shijie Han, Lining Yang, Yuhua Zhou, Xu Cheng, Yibin Wang, and Hongguang Li. 2025a. RankAdaptor: Hierarchical Rank Allocation for Efficient Fine-Tuning Pruned LLMs via Performance Model. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5796–5810, Albuquerque, New Mexico. Association for Computational Linguistics.

Changhai Zhou, Shiyang Zhang, Yuhua Zhou, Qian Qiao, Jun Gao, Cheng Jin, Kaizhou Qin, and Weizhong Zhang. 2026a. AutoQRA: Joint Optimization of Mixed-Precision Quantization and Low-rank Adapters for Efficient LLM Fine-Tuning.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. 2025b. LoRA-drop: Efficient LoRA Parameter Pruning based on Output Evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5530–5543, Abu Dhabi, UAE. Association for Computational Linguistics.

Yuhua Zhou, Changhai Zhou, Shiyang Zhang, Fei Yang, Yi Zhang, and Aimin Pan. 2026b. LaRA: Layer-wise rank allocation for efficient fine-tuning of pruned large language models. *Information Processing & Management*, 63(3):104538.

Mengyao Zhu and Phuc Huu Nguyen. 2026. A Survey of LoRA Algorithm Variations for Language Models. In *Natural Language Processing and Information Systems*, pages 275–290, Cham. Springer Nature Switzerland.

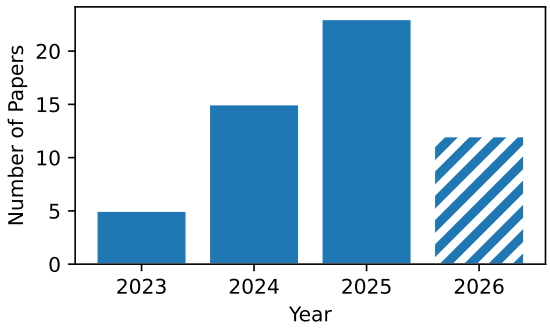


Figure 4: Number of papers (per year) included in the final set.

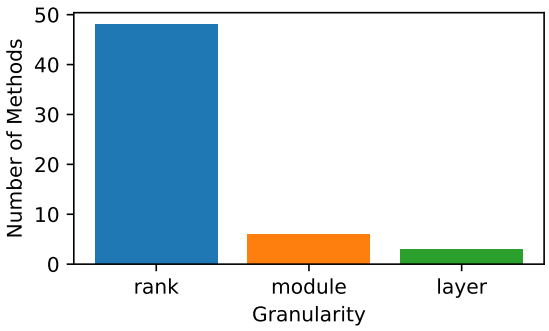


Figure 5: Number of methods per granularity

A Survey Method

We started with an initial set of approximately 20 seed papers identified based on manual search and domain expertise. Using Semantic Scholar's citation graph (Kinney et al., 2025), we expanded the set by screening the complete 1-hop citation neighborhood of each paper (papers cited by, and citing, each paper). Title and abstract were considered during the expansion of the set of papers; newly accepted neighbors were screened until every paper's 1-hop neighborhood had been fully screened and no further papers were added.

After expansion and review, 115 (English-language) were included in the set. We then manually categorized proposed methods, excluding studies that were out of scope or fell below a minimum standard (substantially misrepresenting referenced work or lacking sufficient methodological detail for categorization; 5 papers excluded for quality reasons). In the end, 55 papers remained (see Figure 4). No date, publication-type, or other filters were applied. Non-English papers would have been considered in the search, but none appeared in the final neighborhood.

B Disclosure of Language Model Use in This Work

We used chat-based LLMs to suggest sentence-level edits to check grammar, improve clarity, and shorten paragraphs after a first draft was fully written. All edits were reviewed and verified by the authors. All conceptual contributions, screening, selecting, and summarizing of papers, as well as writing of novel text, were done by the authors without any use of language models.

C Method Scope

We include all methods that produce variable-rank layouts (i.e., excluding methods that may have a dynamic rank during training but at any time use a uniform layout such as `DyLoRA` Valipour et al., 2022)

Further, we exclude methods that modify base weights through iterative low-rank updates. While these methods certainly are interesting, we focus our survey on methods that do *not* edit the base weights, since this enables multi-tenant setups (Chen et al., 2023; Haller et al., 2024; Sheng et al., 2024).

Most of the methods we cover produce truly variable-rank layouts (with $r \in \mathbb{N}$; which was our initial focus), while some select certain modules or layers to fine-tune and assign a fixed rank for those (i.e. $r \in \{0, r_{base}\}$, see Figure 5).

Related, Out-of-Scope Methods

Some methods do lie just outside the scope, sharing similarities with some methods within the scope, but do not produce variable-rank layouts. Some notable examples are discussed in the remainder of this section.

`BiLoRA` (Qiang et al., 2024) uses bi-level optimization to optimize the rank vectors as well as singular values/rank weights. They also experiment with different parameterizations of the singular values to move them towards zero or one. However, they do not test the impact of a final pruning step or report the number of ranks that are actually close to one. So, while the output of this approach is not a variable-rank layout, it could certainly be adapted.

Similarly, `Flexora` (Wei et al., 2025) set module-specific module scaling

`AdapterDrop` (Rücklé et al., 2021) prunes complete adapters from layers below a certain level.

However, the base method does not propose a way to set the threshold.

Wu and Chen (2022) propose pruning complete adapter modules based on the weight magnitudes, but do not apply it to LoRA. The method could be adapted for LoRA and even LoRA ranks and would then be part of the mid-tuning rank pruning group.

DropLoRA (Zhang, 2025) drops out ranks during training in order to make training more robust (like dropout in general Hinton et al., 2012). The dropout rate is constant and not rank-dependent.

Flexora (Wei et al., 2025) selects certain LoRA modules for fine-tuning while freezing others, but does not remove any adapters. No rank-wise selection is performed.

After an initial fine-tuning run where importance scores are collected, PiLoRA (Wang et al., 2025a) splits the output neurons into two sets: A dominant and a supplementary group. For each group, there is a set of ranks (higher rank for the dominant group, lower rank (e.g., 1 or 2) for the supplementary group). Unlike variable-rank methods discussed in this paper, there is no variability in the distribution of the ranks (apart from the difference between the two training stages).

In Tied-Lora (Renduchintala et al., 2024) and VeRA (Kopiczko et al., 2024), the projection matrices are shared across layers (in some cases, they are also frozen). Each layer then has its own u and v scaling vectors (scaling at the level of the intermediate and output representation).

D Variable-rank Methods in Other Taxonomies

Joined Taxonomy. Going with the taxonomy by He et al. (2026), non-LoRA methods such as KronA (Edalati et al., 2022) and Intrinsic-SAID (Aghajanyan et al., 2021) would probably be considered Rank Expansion methods (since Intrinsic-SAID was actually published before the much more well-known LoRA, it should not be considered a LoRA variant, and the name "Reparametrization" should be maintained). We are not aware of any "Reparametrization" approaches that cannot be classified into the LoRA-variant taxonomy by He et al. (2026). Hence, we join their first-level taxonomy groups directly below the "Reparametrization" node.

Variable-rank Approaches in Other Taxonomies. Xu et al. (2023) use the same term to refer to what we call 'variable-rank' approaches. Similarly, Wang et al. (2025b) use the term 'dynamic rank' but do not include any pre-allocation approaches where the rank is allocated prior to training and remains fixed (non-dynamic) during training (pre-allocation methods are not represented within their taxonomy at all). Chen et al. (2025b) use the term 'dynamic low-rank' similarly as well and include DyLoRA (Valipour et al., 2022) which has a dynamic rank during training (i.e., the used rank varies), but does not produce a variable-rank layout.

Yang et al. (2025) include variable-rank methods in their subgroup 'Parameter Efficiency Enhancement $\succ$Parameter Pruning' (which also includes unstructured pruning methods), as well as in group 'Ranking Adaptation $\succ$Rank Refinement' (in which they for example include AdaLoRA (Zhang et al., 2023b) and SoRA (Ding et al., 2023) which we classify as rank pruning methods).

Zhu and Nguyen (2026) include ten methods, which we also include; eight of them under the term 'dynamic selection mechanism'. In this group, they also include LISA (Pan et al., 2024), which only tunes a random subset of the layers during training, but does not prune away any layers during or after training; rsLoRA (Kalajdzievski, 2023) which investigates the impact of the scaling coefficient for LoRA-modules (and suggests to scale the impact by the square root of the rank number instead of the rank number as originally proposed); LoRA+ (Hayou et al., 2024) which derives better learning rates; L-LoRA (Tang et al., 2023), which offers a partial linearization technique for multi-task fine-tuning. Further, they place Light-PEFT and B-LoRA in separate groups.

E Optimizing Memory Use for Dynamic Rank Totals

While post-tuning rank pruning and rank redistribution methods maintain a (relatively) constant total (or average) rank over the course of training, the number of ranks increases in rank growth methods and decreases in mid-tuning rank pruning methods. Since in PEFT settings, the number of trained parameters can have a big impact on the required memory, the total rank can impact how many tokens can be processed in parallel before running out of memory.

However, practical implementations typically use a constant number of tokens during training based on the maximum number of parameters that will be used at some point in training, leaving mem-

ory underutilized in other parts of training. While adding additional complexity, this underutilized memory could potentially be used to process more tokens in that part of training (e.g., increase the batch size or sequence length after pruning ranks mid-tuning). To our knowledge, such an approach has not been explored yet.

F Methods and Design Choices

In the following pages, we include tables detailing the methods included in each subgroup with their respective design choices. While they clearly do not fit into the main part of the section, we consider these a core part of the contribution as they enable researchers to compare existing methods, identify gaps which have not been explored yet, and to select appropriate baselines for specific comparisons.

- **Rank pruning**
  - **Mid-tuning** rank pruning: Table 1
  - **Post-tuning** rank pruning: Table 2
- **Rank growth**: Table 3
- **Rank redistribution**: Table 4
- **Rank pre-allocation**: Table 5
- **Trial-based rank search**: Table 6

Method	♦ granularity	♦ selection criterion	♦ parametrization	♦ orthonormalization	♦ sparsity
`Structured LoRA` (Lawton et al., 2023)	rank	sensitivity-based	LoRA-like	-	one-off mid-training rank removal
`Prune LoRA` (Zhang et al., 2023b)	rank	sensitivity-based	SVD-like	loss-based regularization	iterative mid-training rank removal
`FanLoRA` (Tian et al., 2024)	module	ablation runs without and only with each rank (comparing performance on validation data)	LoRA-like	-	iterative mid-training rank removal
`AutoLoRA` (Zhang et al., 2024)	rank	learned importance values	SVD-like with softmax singular values	-	one-off mid-training rank removal
`Light-PEFT` (Gu et al., 2024b)	rank	sensitivity-based	LoRA-like	-	one-off mid-training rank removal
`LoRPA` (Kim et al., 2024a)	rank	magnitude ($\|\Delta W\|_1$)	LoRA-like	-	one-off mid-training rank removal
`SalientLoRA` (Ke et al., 2024)	rank	salience (influence of singular value on other values combined with orthogonality-aware singular value magnitude)	SVD-like	loss-based regularization	iterative mid-training rank removal
`LoRA-drop` (Zhou et al., 2025b)	module	norm of output activation	LoRA-like	-	one-off mid-training rank removal
`WeightLoRA` (Veprikov et al., 2025)	module	learned module weight	LoRA-like with module-level weights	-	iterative mid-training module removal
`DiffoRA` (Jiang et al., 2025)	module	learned module weights	LoRA-like with module weights	-	one-off mid-training rank removal
`NormAL LoRA` (Aastik et al., 2025)	rank	rank vector norm	LoRA-like	-	one-off mid-training rank removal
`PrunedLoRA` (Yu et al., 2025)	rank	saliency	LoRA-like	-	iterative mid-training rank removal
`SubLoRA` (Gao and Tan, 2026)	rank	Hessian-based	SVD-like	-	iterative mid-training rank removal
`FedARA` (Wu et al., 2026)	rank	Parameter Magnitude	SVD-like	-	iterative mid-training rank removal
`ShapLoRA` (Zhao et al., 2026)	rank	avg. sensitivity over multiple subsets of rank-activations	SVD-like	-	one-off mid-training rank removal

Table 1: Mid-tuning rank pruning methods.

Method	♦ granularity	♦ selection criterion	♦ parametrization	♦ orthonormalization	♦ sparsity
SoRA (Ding et al., 2023)	rank	magnitude of learned gating parameter	SVD-like	-	rank coefficient updated via shrinkage operator
AdaLoRA (Zhang et al., 2023b)	rank	sensitivity-based	SVD-like	loss-based regularization	iteratively setting more singular values to zero
SaLoRA (Hu et al., 2023)	rank	value of learned gating parameter	LoRA-like with Hard-Concrete of gating vector	loss-based regularization	Lagrangian relaxation of constrained optimization ensuring sparsity
LoNAS (Munoz et al., 2024)	rank	val. performance (genetic search)	LoRA-like	-	activation of ranks only below sampled index
ARS (Gao et al., 2024)	rank	binary mask generated by hypernetwork	SVD-like with binary gate/mask generated by hypernetwork	-	regularization of preserved parameters ratio to match target
DoRA (Mao et al., 2024)	rank	frobenius niorm of ΔW_i	SVD-like	no orthonormal-ization, but regularization of component variance	iteratively setting gating values to zero
B-LoRA (Meo et al., 2024)	rank	dropout probability	SVD-like with nested gates/dropout	-	loss-based (minimizing gate activation probabilities)
IST (Yao et al., 2024)	layer	sensitivity-based importance score	LoRA-like	-	selective fine-tuning of complete modules
DPRP (Sharma et al., 2025)	rank	magnitude of singular value	SVD-like	loss-based regularization	compression loss
ARD-LoRA (Shinwari and Usama, 2025)	rank	meta-learned rank scaling factor	LoRA-like	-	loss-based regularization
Drop-LoRA (Chen et al., 2025a)	layer	boundary layer downstream performance	LoRA-like	-	-
VILoRA (Chen and Garner, 2025)	rank	SNR	SVD-like	loss-based regularization	iteratively setting more singular values to zero
CoDyRA (Lu et al., 2026)	rank	learned mask/weight vector	SVD-like	-	rank coefficient updated via shrinkage operator
LoRA-SP (Kim et al., 2026)	rank	thresholding sum of signular values	SVD-like, singular values computed using router	-	loss-based (spectral loss)
RaLo (Deng et al., 2026b)	rank	magnitude of learned singular values	SVD-like	-	rank coefficient updated via shrinkage operator

Table 2: Post-tuning rank pruning methods

Method	♦ granularity	♦ selection criterion	♦ parametrization	♦ orthonormalization	♦ initialization
IncreLoRA (Zhang et al., 2023a)	rank	sensitivity-based (of ΔW_0)	SVD-like	loss-based	A, B normal, singular values zero
GeoLoRA (Schotthöfer et al., 2025)	rank	singular values of expanded coefficient matrix	SVD-like	as part of the update rule (employing SVD)	Initial A and B random but orthonormal, singular values zero. Grown ranks are initialized orthonormal based on data-driven update step.
LoSA (Huang et al., 2025)	rank	reconstruction error (induced by sparsity)	LoRA-like	-	LoRA-like (also for newly added layers)
AROMA (Sheng et al., 2025)	rank	grow until convergence (ratio of last rank norm and remaining weight below threshold)	LoRA-like	-	kaiming init
DR-LoRA (Deng et al., 2026a)	rank	expert routing frequency & sensitivity-based score (with multiplicative aggregation)	LoRA-like MoE	-	LoRA-like
La-LoRA (Gu et al., 2026)	rank	based on rank vector norms	LoRA-like	-	unclear
SeLoRA (Mao et al., 2026)	rank	ratio of Fisher Information between the original and extended ranks	LoRA-like	-	A random, B zero

Table 3: Rank growth methods.

Method	♦ granularity	♦ selection criterion	♦ redistribution mode	♦ parametrization
ALoRA (Liu et al., 2024)	rank	ablation runs without and only with each rank (comparing CE on validation data)	prune bottom k ranks, round-robin module expansion (modules with high scores first)	SVD-like with gating unit
ElaLoRA (Chang et al., 2025)	rank	sensitivity-based	prune bottom k ranks, expand top k modules with the highest minimal values	SVD-like
WeightLoRA+ (Veprikov et al., 2025)	module	learned module weight	prune bottom k ranks,	LoRA-like with module-level weights
L1RA (Singh et al., 2025)	rank	value of gating vector	prune bottom k ranks, round-robin module expansion (modules with high scores first)	SVD-like
FlexLoRA (Liu et al., 2026)	rank	matrix-level entropy of singular values	prune one rank from bottom k modules, expand top k modules	SVD-like

Table 4: Rank redistribution methods.

Method	♦ granularity	♦ information source	♦ selection criterion	♦ initialization
PRILoRA (Benedek and Wolf, 2024)	layer	static	linearly increasing towards higher layers	LoRA-like
RA-LoRA (Kim et al., 2024b)	rank	quantization errors of base weights	normalized cumulative singular values of quantization errors (SVD of error matrix)	LoRA-like
SARA (Gu et al., 2024a)	rank	base weights	sum of singular-values that do not exceed threshold ratio (SVD on base weights)	A, B, singular values are initialized using the normal distribution
AdaRank (Dong, 2024)	rank	hidden states in perturbed model (on target task)	perturbation-induced model disagreements	LoRA-like
GoRA (He et al., 2025)	rank	base weights and gradients (on target task)	sensitivity-based	A random, B scaled Moore-Penrose inverse of A
PLoP (Hayou et al., 2025)	module	hidden states (on target task)	normalize feature norm of activations	LoRA-like
SR-LoRA (Zhang et al., 2025a)	rank	base weights	stable rank (sum of squared singular values of W divided by squared largest singular value)	LoRA-like
EVA (Paischer et al., 2025)	rank	hidden states (on target task)	explained variance (principal directions in input)	A principal directions in inputs, B zero
GeLoRA (Ed-dib et al., 2025)	rank	hidden states (on target task)	2-NN estimate of intrinsic dimension (Facco et al., 2017)	LoRA-like
Sensitivity-LoRA (Zhang et al., 2025b)	rank	diagonal of hessian matrix (on target task)	combination of multiple metrics based on the hessian matrix	LoRA-like
RaLoRA-(Pro) (Ye et al., 2025)	rank	gradients (on target task)	sensitivity-based	MeLoRA-like

Table 5: One-off rank pre-allocation methods.

Method	♦ granularity	♦ search algorithm
RankAdaptor (Zhou et al., 2025a)	rank	proposal based on learned performance model
HyDRA (Xi et al., 2025)	rank	proposal based on learned performance model
AutoQRA (Zhou et al., 2026a)	rank	evolutionary search with Bayesian refinement
LARA (Zhou et al., 2026b)	rank	meta-learned NAS

Table 6: Iterative rank pre-allocation methods.