Designing Large Foundation Models for Efficient Training and Inference: A Survey

Anonymous authors
Paper under double-blind review

Abstract

This paper focuses on modern efficient training and inference technologies on foundation models and illustrates them from two perspectives: model and system design. Model and System Design optimize LLM training and inference from different aspects to save computational resources, making LLMs more efficient, affordable, and more accessible.

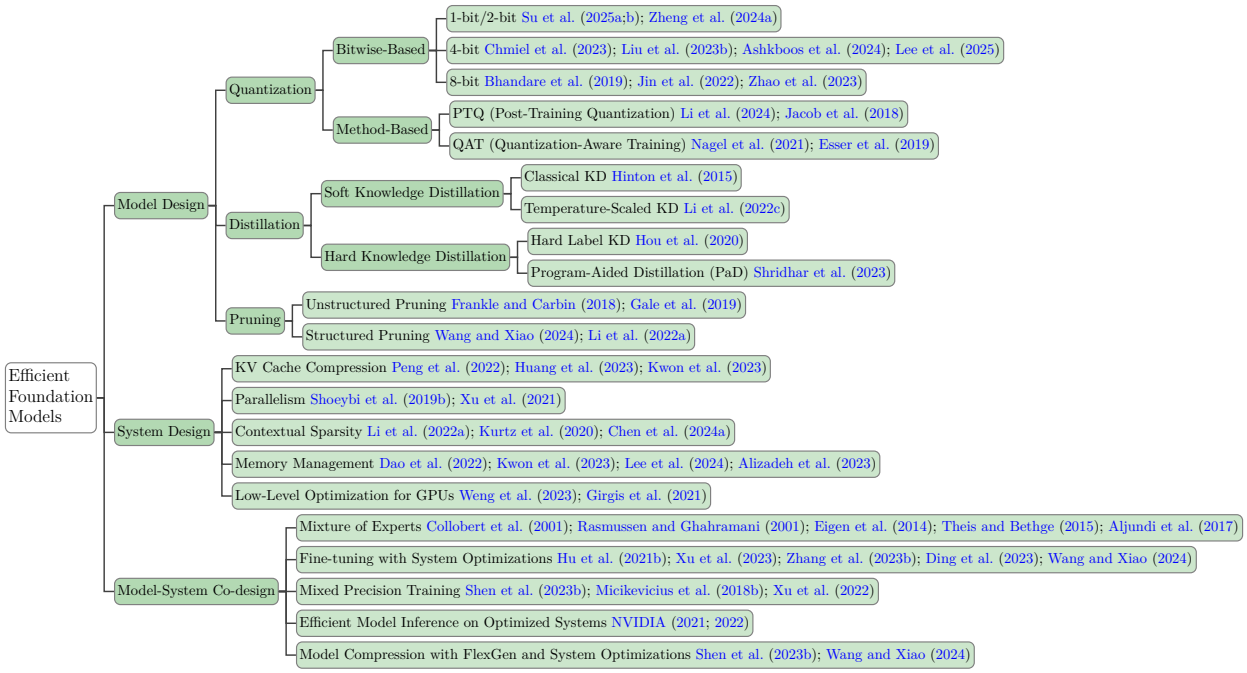

Figure 1: Efficient Foundation Models Overview

1 Introduction

The training and inference processes of foundation models are computationally intensive and require vast amounts of data (Wan et al., 2023; Wang et al., 2024a; Tao et al., 2024; Xiong et al., 2024a). Foundation models include multimodal models capable of handling diverse data types such as text, images, and audio. According to statistics (Bubeck et al., 2023), training models like GPT-4 can take 4–7 months, requiring thousands of GPUs, and results in models with billions of parameters. Such a high demand for computational resources, combined with the large size of the resulting models, makes a barrier for many researchers and organizations who want to train large language models but lack the financial resources to cover the computation expense. It also poses challenges for individuals who want to deploy such models on personal computers or mobile devices but lack sufficient local computational resources. Given these challenges, we

anticipate that efficient design for foundation models will become a major focus in the next wave of AI advancements.

Efficient design for foundation models can generally be approached from three key perspectives: **Model Design**, **System Design**, and **Model-System Co-Design**.

2 Model Design for Efficient Foundation Models

2.1 Quantization

Quantization is a technique that reduces the numerical precision of model parameters and activations to improve computational efficiency. By representing values with lower-bit data types, it significantly reduces memory usage and accelerates inference. This is particularly important for deploying large-scale foundation models on resource-constrained hardware. In general, quantization methods can be categorized into **bit-width based quantization** and **method-based quantization**. Bit-width based quantization lowers precision using formats like INT8, FP8, and even lower-bit representations such as 4-bit and 2-bit. These methods apply scaling factors, Hessian-aware adjustments, and structured weight transformations to minimize accuracy loss while improving efficiency. Floating-point quantization, such as FP8, further enhances adaptability by handling activations with large dynamic ranges. Method-based quantization includes Post-Training Quantization and Quantization-Aware Training. PTQ quantizes a pre-trained model without modifying its weights, enabling fast deployment with minimal computational cost. In contrast, QAT incorporates quantization into the training process, allowing models to adapt to lower precision. Additionally, mixed-precision quantization assigns different bit-widths to different layers - it dynamically assigns bit-widths per layer to balance efficiency and accuracy, leveraging advanced techniques like adaptive scaling, reordering transformations, and structured low-rank decomposition.

By leveraging these techniques, quantization enables efficient large-model training and inference while preserving performance. The following sections explore these strategies in more detail.

2.1.1 Bit-width Based Quantization

Bit-width based quantization reduces numerical precision in model parameters and activations, optimizing memory usage and computational efficiency. 8-bit quantization methods, such as GPTQ (Frantar and Alistarh, 2022) and SmoothQuant (Xiao et al., 2023), primarily rely on fixed-point representations like INT8 and FP8 to maintain numerical stability. The standard quantization function follows $q(x) = \text{round}(x/s)$, where s is a learned scale factor. FP8 methods such as F8Net (Dettmers et al., 2023) introduce fixed-point multiplication, eliminating INT32 overhead via dyadic scaling $q(x) = \text{round}(x \cdot 2^{-p})$, where p dynamically minimizes quantization error. Compared to INT8, FP8 formats such as E5M2 and E4M3 have a wider dynamic range, adapting better to activations with large outliers, leading to improved numerical stability in LLMs and Vision Transformers (Shen et al., 2024).

Low-bit quantization (4-bit) approaches such as GPTQ (Frantar et al., 2022b) and AWQ (Huang et al., 2024) apply per-layer Hessian-aware weight quantization to minimize accuracy degradation, solving $\min_{\hat{W}} \|W - \hat{W}\|_H^2$, where H is the Hessian capturing weight sensitivity. Atom (Zhao et al., 2024b) further refines this with mixed-precision weight-activation quantization, applying per-cluster scale factors $Q(w) = \text{round}(w/s_h)$ for high-magnitude weights and $Q(w) = \text{round}(w/s_l)$ for low-magnitude ones. Additionally, DilateQuant (Zhang et al., 2023f) introduces weight dilation, defining a dilation factor s_{dilate} such that $\tilde{W} = (W/\max|W|) \cdot s_{\text{dilate}}$, ensuring stable quantization across weight ranges. Hybrid 4-bit formats, such as NF4 and FP4, employ logarithmic distributions to preserve high-magnitude weight precision while compressing low-importance activations.

For extreme quantization (2-bit and 1-bit), BiDM (Zheng et al., 2024b) employs Timestep-friendly Binary Structure (TBS), applying the binary activation function $A_b = \text{sign}(A)$ with a straight-through estimator (STE) for gradient propagation, $\frac{\partial L}{\partial A} \approx \frac{\partial L}{\partial A_b}$. RotateKV (Su et al., 2025a) mitigates degradation by applying rotation-based quantization transformations $\tilde{W} = RWR^T$, where R is an adaptive rotation matrix computed

via Fast Walsh-Hadamard Transform (FWHT). A generalized formula for extreme quantization using mixed-binarization strategies follows:

$$q(x) = \arg\min_{q \in Q} \|x - q\|^2 + \lambda \sum_j C(q_j), \tag{1}$$

where $C(q_j)$ accounts for quantization constraints across different bit-widths. This formulation captures a wide range of low-bit strategies.

2.1.2 Method-Based Quantization

Post-Training Quantization (PTQ) methods quantize models after training, offering fast deployment without retraining. GPTQ (Frantar et al., 2023) leverages second-order Hessian-aware loss minimization, solving $\min_{\hat{W}} \|W - \hat{W}\|_H^2$. PTQD (Wu et al., 2023b) introduces variance calibration, modeling quantization noise as $\Delta_{\text{quant}} = \hat{\mu} - \mu + \sigma\epsilon$, where μ and σ are activation statistics, and ϵ is Gaussian noise. Reorder-based PTQ (RPTQ) (Yuan et al., 2023) refines layer-wise quantization by applying reordering matrices P and Q, reformulating weights as $\tilde{W} = PWQ^{-1}$, effectively reducing outlier sensitivity. FP8-based PTQ (Shen et al., 2024) exploits dynamic scaling, adjusting e_{scale} in $Q(x) = \text{round}(x \cdot 2^{e_{\text{scale}}})$ per layer, ensuring uniform activation distribution across Transformer layers.

Quantization-Aware Training (QAT) incorporates quantization during training, allowing adaptation to low-bit constraints. EfficientDM (Yang et al., 2023) fine-tunes diffusion models with quantization-aware low-rank adapters (QALoRA), optimizing $Y = \text{QU}(X, s_x) \cdot \text{QU}(W + BA, s_w)$, where B and A are low-rank matrices preserving accuracy. DoReFa-Net (Zhou et al., 2016) utilizes logarithmic quantization $q(x) = 2^{\lfloor \log_2(|x|) \rfloor} \text{sign}(x)$, preventing information loss at ultra-low bit widths. Q-Diffusion (Li et al., 2023a) employs per-timestep mixed-precision strategies, dynamically adjusting s_t in $Q(X_t) = \text{round}(X_t/s_t) \cdot s_t$, stabilizing diffusion process quantization.

Mixed-Precision Quantization dynamically adjusts bit-width allocations per-layer to balance efficiency and accuracy. MPQ-DM (Wu et al., 2023b) introduces per-layer mixed-precision assignment via $q(x, b) = \text{round}(x/s_b)$, where s_b is optimized for different bit-widths. Quant-LLM (Yao et al., 2023) employs FP6-centric quantization, formulating an optimal bit allocation problem:

$$\min_{s_b} \sum_i \left(x_i - Q(x_i, b)\right)^2 + \lambda \sum_j C(b_j), \tag{2}$$

where $C(b_j)$ represents computational cost constraints. DilateQuant (Zhang et al., 2023f) extends weight dilation techniques, ensuring adaptive scaling across Transformer layers. Adaptive Quantization strategies such as RotateKV (Su et al., 2025a) compress KV caches via rotation-based transformations $\tilde{K} = RK, \quad \tilde{V} = RV$, minimizing information loss. PackQViT (Dong et al., 2023) optimizes vision models using logarithmic quantization $Q(x) = 2^{\lfloor \log_2(x) \rfloor}$, further improving bit precision trade-offs.

2.2 Knowledge Distillation

Knowledge Distillation (KD) has emerged as a fundamental strategy in optimizing large language models (LLMs) for deployment in environments with limited computational resources. As LLMs increase in complexity and size to achieve high performance across diverse language tasks, distillation techniques provide a means to compress these models while retaining essential capabilities. In the context of LLMs, KD is divided into **Soft Knowledge Distillation** and **Hard Knowledge Distillation**, each playing a unique role in balancing model size, accuracy, and efficiency. Soft distillation enhances LLM generalization by transferring nuanced probability distributions, while hard distillation prioritizes specific label accuracy and simplifies training. These techniques are pivotal in the ongoing development of compact yet capable LLMs, allowing them to retain high performance even when scaled down significantly.

2.2.1 Soft Knowledge Distillation

Soft distillation leverages the full probability distribution generated by an LLM teacher model, capturing subtleties in the relationship between various output classes. For LLMs, this means the student model not only learns the correct answers but also internalizes the nuances in the teacher's responses to a broad array of inputs. By transferring these probability distributions, soft distillation helps the student model mimic the teacher's language understanding, sentiment detection, and reasoning across tasks. Applying temperature scaling to soften the teacher's outputs, the student LLM can absorb richer linguistic and semantic representations, making it particularly effective for LLMs intended for general-purpose language understanding. As a result, soft distillation is integral for developing efficient LLMs that retain robust performance across diverse tasks.

Probability Distribution Probability distribution methods involve transferring the full output probabilities from the teacher model to the student. This helps the student capture nuances in class relationships, learning not only the correct answers but also the likelihood of various alternatives. The choice of divergence function of probability distribution in KD shapes the way the student model aligns with the teacher's output distribution, impacting both diversity and quality of generated content. A commonly used approach is forward Kullback-Leibler (KL) divergence, which minimizes the divergence by aligning the student distribution to cover all modes of the teacher distribution, as seen in work by Hinton et al. (2015), Romero et al. (2015) and Zagoruyko and Komodakis (2017). This method ensures that the student model captures the full spectrum of the teacher's probability distribution, even for tokens assigned low probability by the teacher. However, this "mode-covering" approach may lead the student to overestimate these low-probability regions, potentially resulting in hallucinations or lower-quality generations. Alternatively, reverse KL divergence focuses on aligning the student with only the high-probability regions of the teacher distribution, as demonstrated by Gu et al. (2024b). However, reverse KL can come at the expense of diversity, often producing fewer output variations. In Baby LLaMA, temperature scaling is applied to the logits of teacher models (e.g., GPT-2 and LLaMA) to provide smoother output distributions for distillation Timiryasov et al. (2023). This allows the student model to retain the nuances of the teacher's probabilistic predictions, especially in low-data regimes. The soft probabilities are controlled by the temperature parameter T, ensuring that the model retains fine-grained knowledge:

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}.$$

Expanding on this, KPTD adopts a similar approach, but it leverages entity definitions and domain-specific knowledge to generate the transfer set. The student model learns to match the teacher's output distribution based on this enriched set of entities, which is crucial for maintaining up-to-date knowledge in pre-trained models Mirzadeh et al. (2020); Gong et al. (2022); Chen et al. (2020). Following this, the authors of GKD refine this further by using token-level forward and reverse divergence, focusing on critical tokens during distillation Jafari et al. (2021); Kim and Rush (2016). This design ensures that the student captures vital aspects of sequence-level tasks, such as machine translation and summarization, where token importance fluctuates throughout the sequence.

In parallel with previous work, in MetaICL Min et al. (2022) and Multitask-ICT Huang et al. (2022), soft distillation is adapted to transfer in-context learning (ICL) abilities from large models to smaller ones, helping the student generalize across tasks. These models use few-shot learning to transfer the multitask capabilities of large LLMs, effectively allowing the student to leverage the soft output distributions for task generalization. After this, the work from Zhao et al. (2024a) further enhances this process by introducing a multi-stage collaborative approach, where the student generates refined pseudolabels to improve the next stage of distillation, pushing the boundaries of semi-supervised learning through iteration.

Above all, the soft distillation loss can be expressed as:

$$\mathcal{L}_{\text{distill}} = \lambda_{\text{fwd}} \sum_{i=1}^{N} \alpha_i \cdot D_{\text{KL}}\left(p_{\text{teacher},i} \| p_{\text{student},i}\right) + \lambda_{\text{rev}} \sum_{i=1}^{N} \beta_i \cdot D_{\text{KL}}\left(p_{\text{student},i} \| p_{\text{teacher},i}\right),$$

$$D_{\mathrm{KL}}(p\|q) = \sum_j p_j \log \frac{p_j}{q_j}$$

where:

- $p_{\text{teacher},i}$ and $p_{\text{student},i}$ are the teacher and student probability distributions for token i, respectively.
- λ_{fwd} and λ_{rev} are tunable weights balancing forward and reverse KL components.
- α_i and β_i are token-specific importance weights, which could depend on token-level saliency, attention scores, or domain-specific criteria.
- N is the total number of tokens in the sequence.

2.2.2 Hard Knowledge Distillation

Hard distillation simplifies the training process by focusing solely on the teacher model's "hard" labels, or binary outputs, for each task. In the case of LLMs, hard distillation emphasizes specific target outcomes over nuanced inter-class information, which can be advantageous for task-focused LLMs where interpretability and task accuracy are prioritized. This method is often used in classification tasks where the LLM's primary purpose is to deliver accurate, targeted answers without requiring complex linguistic or semantic generalization. Although this approach may limit the student's understanding of intricate language structures, it accelerates training and achieves effective LLM compression, making it a suitable choice for specialized language tasks where efficiency and specific label accuracy are crucial.

Program-aided Distillation (PaD) addresses the challenge of faulty reasoning in distillation by introducing programmatic reasoning Shridhar et al. (2023). Synthetic programs are generated to ensure correct reasoning steps, and these programs are automatically checked by an interpreter before being used in distillation. This reduces faulty reasoning in student models, ensuring that the correct reasoning paths are learned alongside the correct outputs. DynaBERT applies hard knowledge distillation in dynamic model compression, progressively decreasing model size while ensuring that smaller models retain the performance of the teacher Hou et al. (2020). By combining hard distillation with a reduction in both model width and depth, DynaBERT balances efficiency with accuracy. Similarly, LaMini-LM focuses on transferring instruction-following abilities to smaller models by using hard distillation on a curated instruction set, optimizing the student for task-specific performance Wu et al. (2023a). Zephyr integrates hard knowledge distillation with Direct Preference Optimization (dDPO) Tunstall et al. (2023). In this approach, the student learns from both hard decisions and user preferences, blending rigid output labels with nuanced optimization for user alignment. This hybrid approach is particularly effective in conversational AI, where both categorical responses and preference-based feedback are important for generating relevant and engaging responses.

Above all, the hard distillation process can be formalized as :

$$\mathcal{L}_{\text{hard}} = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}^{\text{teacher}} \cdot \log p_{i,j}^{\text{student}},$$

where:

- $y_{i,j}^{\text{teacher}} \in \{0,1\}$ is the one-hot encoded hard label for token i and class j, generated by the teacher model.
- $p_{i,j}^{\text{student}}$ is the predicted probability of the student model for token i and class j, computed using a softmax over the student logits:

$$p_{i,j}^{\text{student}} = \frac{\exp(z_{i,j}^{\text{student}})}{\sum_{k=1}^{C}\exp(z_{i,k}^{\text{student}})},$$

where $z_{i,j}^{\text{student}}$ is the logit for class j at token i.

- N is the total number of tokens in the dataset.
- C is the number of classes in the classification task.

2.3 Pruning

In addition to quantization and knowledge distillation, pruning is another efficient technique for reducing computational costs in LLMs. While quantization focuses on hardware optimization and knowledge distillation on knowledge transfer in neural networks, pruning concentrates on the model's structural efficiency. Specifically, pruning in machine learning involves removing less important connections in the neural network while retaining the important ones, making the model more memory-efficient and faster.

Pruning methods can be broadly categorized based on their granularity into unstructured pruning and structured pruning. Structured pruning removes entire neurons or larger components of the neural network. In contrast, unstructured pruning refers to the removal of individual connections, as seen in the conventional pruning methods studied by Han et al. (2015). Additionally, semi-structured pruning like N:M sparsity (Mishra et al., 2021), is generally considered a specialized form of unstructured pruning. In general, structured pruning usually yields better inference speed improvements compared to unstructured pruning, due to its hardware-friendly design.

Pruning methods for LLMs typically remove weights based either on (1) the loss function or (2) the layer output.

(1) Loss-based. The weights are usually removed by computing metric such as gradients with respect to the training or validation loss. In some cases, pruning is performed directly through optimization techniques that minimize the model's loss after weight removal.

(2) Layer output-based. Alternatively, other pruning methods for LLMs aim to find the following masks in a layer-wise pattern. The pruning quality is usually measured by the ℓ_2-norm between the output, for given specific inputs X_ℓ, of the uncompressed layer and that of the compressed one. More formally, consider a layer ℓ with weight matrix $W_\ell \in \mathbb{R}^p$. The objective is to find the binary mask $M_\ell \in [0,1]^p$ and possibly an updated weight matrix $\hat{W}_\ell$ so that the following objective is minimized,

$$\arg\min_{M_\ell, \hat{W}_\ell} \|W_\ell X_\ell - (\hat{W}_\ell \odot M_\ell) X_\ell\|_2^2. \tag{3}$$

Here, $\odot$ denotes element-wise multiplication.

2.3.1 Unstructured Pruning

In this section we consider unstructured pruning and semi-structured pruning methods like N:M sparsity.

Saliency-Based Pruning. There are several works motivated by classical post-training methods, including Optimal Brain Damage (OBD; LeCun et al. (1989)) and Optional Brain Surgeon (OBS; Hassibi and Stork (1992)). In the original OBS framework, the removal of connection at index m[1] with weight w_m is selected based on the saliency metric

$$S_m = \frac{W_m^2}{[H^{-1}]_{mm}},$$

where H is the Hessian matrix.

[1]Note that we use subscript S_{ij} and S_m interchangeably. S_{ij} emphasizes the weight within a specific matrix, while S_m emphasizes the weight across the entire neural network.

After the weight removal, all the remaining weights will be updated with the following formula:

$$\delta_m = -\frac{w_m}{[H^{-1}]_{mm}}[H^{-1}]_{:,m}$$

to compensate for the removal of the weight at index m.

Note that under the formulation of Equation 3 for an affine transformation, each row can be computed independently. Specifically, we can iteratively apply OBS to different rows by (1) identifying the location m that results in the smallest loss increase, (2) constructing the updated mask $M - \{m\}$, and (3) updating **all** remaining weights responding to $M - \{m\}$ upon the removal of mask $\{m\}$.

Consider a weight matrix $W_\ell \in \mathbb{R}^{d_{row} \times d_{col}}$, SparseGPT (Frantar and Alistarh, 2023) extends the idea of OBS by updating only a subset of remaining weights $H \subseteq M - \{m\}$ to enable faster computation of the Hessian matrix. More specifically, SparseGPT processes columns sequentially from left to right using a sequence of Hessian matrices, which can be efficiently computed using Gaussian elimination. SparseGPT only uses the weights to the right of the pruned weight as subset to compensate the loss induced by its removal. The computational cost of SparseGPT is hence $\Theta(d^3_{hidden})$ compared to $\Theta(d^4_{hidden})$ required by exact construction in Transformers with hidden dimension d_{hidden}.

SparseGPT is also compatible with N:M sparsity by choosing N smallest weights that incurs the lowest errors in every block of M. Additionally, it also supports quantization method like GPTQ (Frantar et al., 2022a).

Wanda (Sun et al., 2023a) proposes a simple and effective pruning metric:

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2,$$

where W is a weight matrix, and X_j is the j-th feature normalized by batch and sequence length dimensions. Wanda shows the connection between its pruning criterion and those of OBS and SparseGPT through a diagonal approximation of SparseGPT's saliency score:

$$S_{ij} = \left[\frac{|W|^2}{\text{diag}((XX^T + \lambda I)^{-1})}\right]_{ij}.$$

Applying the diagonal approximation with $\lambda = 0$, the metric can be simplified to:

$$\left[\frac{|W|^2}{\text{diag}((XX^T + \lambda I)^{-1})}\right]_{ij} \overset{\lambda=0}{\approx} \left[\frac{|W|^2}{(\text{diag}(XX^T))^{-1}}\right]_{ij} = (|W_{ij}| \cdot \|X_j\|_2)^2. \tag{4}$$

Compared to the complexity $\Theta(d^3_{hidden})$ of SparseGPT, Wanda enjoys the complexity of $\Theta(d^2_{hidden})$. Additionally, Wanda does not need to update the weights.

Empirical evaluations on zero-shot tasks and perplexity on LlaMA (Touvron et al., 2023a) models show the competitive performance of Wanda compared to SparseGPT. Wanda is also compatible with semi-structured N:M pruning and LoRA fine-tuning (Hu et al., 2021c).

Improved Saliency Criterion (ISC; (Shao et al., 2024)) proposed to combine criterion from both OBD and OBS to define a better saliency score:

$$S_m = W_m^2 \left(H_{mm} + \frac{1}{[H^{-1}]_{mm}}\right).$$

Moreover, ISC also explores the effects of the sparsity ratio allocations between layers through sensitivity. The sensitivity is calculated by the average trace of the Hessian matrix. Experiments conducted on the LlaMA model family and Baichuan models show its strong performance and compatibility with quantization methods.

D-Pruner (Zhang et al., 2024c) studies domain-specific compression for LLMs. It uses the formulation of OBS for general knowledge and while introducing additional regularization term tailored to domain-specific

knowledge. With the help of both general and domain-specific calibration datasets, D-Pruner produces a pruned model that strikes a balance between both generality and specificity.

RIA (Zhang et al., 2024d) incorporates relative importance into Wanda's saliency score, making the final saliency score:

$$S_{ij} = \left(\frac{|W_{ij}|}{\sum_i |W_{ij}|} + \frac{|W_{ij}|}{\sum_j |W_{ij}|} \right) \cdot (\|X_i\|_2)^{\alpha}. \tag{5}$$

Moreover, RIA introduces an optimization specifically for N:M sparsity through channel permutation. This enables the retention of important weights within the same block, improving the performance of pruned models.

Pruner-Zero (Dong et al., 2024) formulates pruning as Symbolic Regression (SR) problem and use genetic programming (GP) to solve it. It directly searches pruning saliency score using LLM statistics as input features, including activations (X), gradients (G), and weights (W). The search space includes primitive operations include unary and binary ones. Pruner-zero iteratively generates and refines pruning metrics via genetic programming procedures including tournament selection, subtree crossover, and node mutation.

There are other studies focus more on deployment and evaluation rather than solely on algorithm design. For example, Prune-And-Tune (Syed et al., 2023) demonstrates that fine-tuning can further improve the performance of SparseGPT for OPT model family, particularly at high sparsity levels. Shamrai (2024) specifically evaluates SparseGPT and Wanda specifically for the Ukrainian language.

Optimization Based Pruning. Another line of research aims at finding masks through optimization.

BESA (Xu et al., 2024) efficiently learns block-wise sparsity ratios for each model component/block, such as Transformer blocks and feed-forward network (FFN) blocks, through minimizing the reconstruction error. The training process uses a standard Straight-through estimator (STE) to handle the non-differentiability of the masks. The sparsity is enforced through a ℓ_2 regularization term during the training process.

Once the sparsity ratios are learned, Wanda saliency score will be used to prune unimportant weights within each block. Additionally, BESA is also compatible with quantization techniques.

2.3.2 Structured Pruning

Saliency-based Pruning. Ma et al. (2023) highlights that task-specific model compression through knowledge distillation can be time-consuming in some cases, which requires up to 3.5 days on 4 GPUs for a 20GB corpus. In contrast, pruning can significantly compress model size on a structural level, which requires up to 3 hours on 1 GPU for a 50MB corpus after pruning (Wang et al., 2024b).

One famous method of pruning in LLMs is LLMPrunner (Ma et al., 2023). Initially, a dependency tree or graph is established based on structural dependency in LLMs, linking neurons based on dependency relationships. After building the dependency graph, the pruner then analyzes the importance of each link in the graph and trims those links with less importance to save computational resources and accelerate inference. LLMPrunner uses multiple methods (element-wise, vector-wise, group) to evaluate the importance of coupled structures in the dependency graph, removing less important structures. It also offers reliable recovery methods such as Low-rank Approximation to recover links discarded by mistake. Empirical results show that LLMPrunner can achieve 20% parameter reduction while retaining 90% of the original performance, showcasing the effectiveness of pruning in maintaining model efficiency while reducing computational resources by removing redundant elements in the network.

FLAP (An et al., 2024) measures the importance of different channels using fluctuation-based metrics with calibration data. It then normalizes these metrics to to ensure a consistent comparison of scores across different layers and modules. Additionally, FLAP introduces a technique called baseline bias compensation to deal with the damage caused by removing components.

Zhang et al. (2024a) proposes a structured pruning method that groups kernels and features independently and evaluates their importance by second-order Taylor expansion. To reduce computation, they approximate

the second order term with squared gradient. Specifically, for kernel group C, the saliency score S_C is computed as:

$$\left|\sum_{c\in C}\frac{\partial\mathcal{L}}{\partial c}c-\frac{1}{2}\left(\frac{\partial^2\mathcal{L}}{\partial c^2}c^2\right)\right|\approx\left|\sum_{c\in C}\frac{\partial\mathcal{L}}{\partial c}c-\frac{1}{2}\left(\frac{\partial\mathcal{L}}{\partial c}c\right)^2\right|=S_C,$$

where $\mathcal{L}$ denotes the loss function and c represents the parameters in the group C.

Similarly, for features f, the corresponding saliency score of an affine transformation is

$$S_f=\sum_C\left|\sum_{c\in C[:,f]\cup C[f,:]}\frac{\partial\mathcal{L}}{\partial c}c-\frac{1}{2}\left(\frac{\partial\mathcal{L}}{\partial c}c\right)^2\right|.$$

Optimization-based Pruning. ShearedLlaMA (Xia et al., 2023b) formulates the pruning problem as a constrained optimization problem using Lagrange multiplier. It learns pruning masks at varying granularities, from coarse ones like entire layers and hidden dimensions to fine-grained ones like attention heads and intermediate dimensions.

Encoder-Decoder Architecture. NASH (Ko et al., 2023) specifically studies structured pruning for encoder-decoder Transformer architecture. Inspired by empirical findings, NASH proposes to shorten, i.e., reduce depth, for decoders and narrow, i.e., reduce width, for encoders. To achieve better performance, NASH additionally applies hidden states distillation by adding a Kullback–Leibler (KL) divergence loss term.

Coupled with PEFT. Light-PEFT (Gu et al., 2024a) and V-PETL (Xin et al.) aims to reduce the costs of fine-tuning at multiple granularities. At a coarse level, it measures the importance of PEFT modules through the change of PEFT modules on the original module output. At a finer level, it measures the rank of each PEFT modules by summing the first-order Taylor expansion saliency score of all parameters of that rank:

$$S_{i,j}=\left|\frac{\partial\mathcal{L}}{\partial W_{i,j}}W_{i,j}\right|.$$

Similarly, LoRAPrune (Zhang et al., 2024b) uses the same Taylor expansion saliency score for measuring group importance. It inserts learnable LoRA matrices during downstream task adaption, circumventing the needs of computing the gradients of the entire weight matrix.

2.3.3 Others

Infra. FlashLLM (Xia et al., 2023a) offers specific infrastructure support for unstructured pruning by utilizing Tensor Cores rather than SIMT Cores to enable faster inference.

3 System Design for Efficient Foundation Models

Large Foundation Models face significant challenges in managing computational and memory costs during autoregressive generation. Effective system design leverages optimizations like Key-Value (KV) caching, token reduction, and precision scaling to balance efficiency and accuracy. These methods are grounded in system design that formalize the trade-offs between model performance and resource utilization.

3.1 KV Cache Design

KV caching is a cornerstone of transformer-based LLMs, significantly reducing computational overhead by reusing previously computed key ($\mathbf{K}$) and value ($\mathbf{V}$) matrices. During autoregressive generation, the self-attention mechanism computes the output for a sequence of tokens by leveraging the precomputed cache.

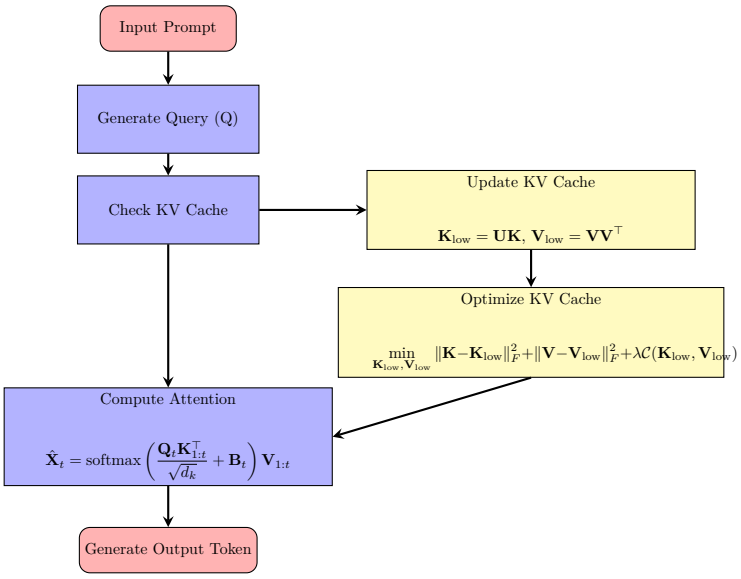

Figure 2: KV Cache Design Pipeline

Let $\mathbf{Q}_t$, $\mathbf{K}_{1:t}$, and $\mathbf{V}_{1:t}$ denote the query, key, and value matrices at time t. The attention mechanism scales the query-key interaction by the inverse square root of the key dimension d_k, applies positional biases $\mathbf{B}_t$, and generates the final representation $\hat{\mathbf{X}}_t = \text{softmax}((\mathbf{Q}_t\mathbf{K}_{1:t}^\top)/\sqrt{d_k} + \mathbf{B}_t)\mathbf{V}_{1:t}$. To improve memory efficiency, advanced KV cache compression methods use low-rank approximations, where keys and values are projected into lower-dimensional spaces using learned projection matrices $\mathbf{U}$ and $\mathbf{V}$. Specifically, the compressed representations are given by $\mathbf{K}_{\text{low}} = \mathbf{UK}$ and $\mathbf{V}_{\text{low}} = \mathbf{VV}^\top$, where $r \ll d_k$ controls the rank of the approximation. Query-aware sparsity further optimizes KV caching by dynamically identifying and updating only the most relevant tokens using a sparsity mask $\mathbf{M}_t$, defined as the top-k entries of the scaled query-key scores. This hierarchical approach combines memory reduction and computational efficiency while maintaining the fidelity of attention outputs.

The general optimization for KV cache design balances memory usage, computational cost, and attention fidelity, expressed as:

$$\min_{\mathbf{K}_{\text{low}},\mathbf{V}_{\text{low}}} \|\mathbf{K} - \mathbf{K}_{\text{low}}\|_F^2 + \|\mathbf{V} - \mathbf{V}_{\text{low}}\|_F^2 + \lambda \cdot \mathcal{C}(\mathbf{K}_{\text{low}}, \mathbf{V}_{\text{low}}),$$

where $\mathcal{C}(\mathbf{K}_{\text{low}}, \mathbf{V}_{\text{low}})$ models the memory and computational costs of the compressed representations (Huang et al., 2023; Xiong et al., 2024b).

Autoregressive decoding in LLMs requires KV cache, which stores intermediate outputs for all tokens, creating significant memory overhead. Some adaptive KV cache compression techniques will adjust the precision of stored tokens based on their importance. Let $K, V \in \mathbb{R}^{T\times d}$ represent the key and value matrices. The optimization for adaptive precision is:

$$\min_{\hat{K},\hat{V}} \|K - \hat{K}\|_F^2 + \|V - \hat{V}\|_F^2 + \gamma \cdot \sum_{t=1}^{T} P_t \cdot \mathcal{C}(P_t),$$

where $\hat{K}$ and $\hat{V}$ are the compressed representations, P_t denotes the precision level for token t, and $\mathcal{C}(P_t)$ quantifies the cost of using a specific precision. Precision levels can be dynamically assigned as $P_t = \text{argmax}_{p\in\{\text{FP32,FP16,FP8,INT8},etc\}}\left\{S_t \cdot \text{Utility}(p)\right\}$, where S_t is the token saliency score and $\text{Utility}(p)$ reflects the precision's effectiveness. VL-Cache achieves up to 90% memory reduction with a 7× improvement in decoding latency (Tu et al., 2024).

3.2 Token Reduction

Token reduction focuses on minimizing the number of tokens processed during training and inference while retaining essential information for accurate predictions. In transformer models, tokens often exhibit redundancy, particularly in the deeper layers where contextualized embeddings converge. Let $T \in \mathbb{R}^{n \times d}$ denote the token embeddings, where n is the token count and d is the embedding dimension. Token pruning techniques dynamically identify and remove less important tokens based on learned importance scores or predefined heuristics. Importance scores can be computed using attention weights, token magnitudes, or cumulative relevance across layers. For example, attention-based pruning ranks tokens by their cumulative importance $S_i = \sum_{l=1}^{L} \sum_{j=1}^{n} A_{ij}^{(l)}$, where $A^{(l)}$ represents the attention matrix at layer l. Tokens with the lowest scores are removed, and their information is redistributed to the remaining tokens. Alternatively, token merging methods combine similar tokens, reducing the total count while preserving overall representation. In methods like Token Merging (ToMe), merged tokens are computed as weighted combinations, where similarity scores guide the merging process. The merged embeddings $\hat{T}_k = \alpha T_i + (1 - \alpha) T_j$, with α determined by cosine similarity, ensure that merged tokens retain meaningful features. These approaches effectively reduce computational costs in both text and visual transformer architectures.

The optimization objective for token reduction can be formulated as:

$$\min_{M, \hat{T}} \|T \odot M - \hat{T}\|_F^2 + \mu \cdot \text{Cost}(M) + \nu \cdot \text{Regularization}(M),$$

where $M \in \{0, 1\}^n$ is the token retention mask, $\text{Cost}(M)$ captures the computational overhead, and $\text{Regularization}(M)$ enforces structural constraints (Bolya et al., 2023; Rao et al., 2021).

For visual LLMs, visual tokens encode rich spatial and semantic information but often contain redundancy. PyramidDrop adopts token reduction framework by computing redundancy scores for each token based on local similarity metrics (Xing et al., 2024). Similarly, Victor leverages token summarization mechanisms, where token embeddings are aggregated into compact registers to reduce computation by over 3× with minimal accuracy loss (Wen et al., 2024).

3.3 Sparsity-Aware Optimization

The QKV transformation requires masking for autoregressive prediction. The structure $QK^\top \odot M$ results in a lower triangular masked attention matrix of size $N \times N$, where N is the processed sequence length, which can be significantly large in large-scale language models.

However, not all tokens contribute equally to the final output. Studies show that a small subset of tokens dominates attention computations, while others carry minimal impact. Optimizing attention sparsity reduces computational overhead without sacrificing performance. Several methods address this:

Token Selection and Structured Sparsity Some Research reduces computational costs by focusing only on high-saliency tokens. Let $A \in \mathbb{R}^{T \times T}$ represent the attention matrix at a given layer, where A_{ij} denotes the attention weight between tokens i and j. Sparsity-aware optimization selects a subset of tokens $S = \{t \mid S_t > \tau\}$, where $S_t = \sum_{j=1}^{T} A_{tj}$ represents the token's saliency score and τ is a dynamically determined threshold. The optimization problem can be expressed as:

$$\max_{\rho_l} \sum_{l=1}^{N_L} \rho_l \cdot \sum_{t \in S} \log\left(1 + \frac{S_t}{\tau}\right) - \kappa \cdot \mathcal{C}(\rho_l),$$

where ρ_l denotes the retention ratio at layer l, $\mathcal{C}(\rho_l)$ is the computational cost associated with retaining tokens, and κ balances performance and cost. Methods like ZipVL and H_2O dynamically adjust ρ_l across layers based on the attention distribution (He et al., 2024; Zhang et al., 2023e), achieving over large memory and cache savings with small accuracy degradation.

Activation and Model Weight Sparsity Beyond attention sparsity, recent studies have explored the natural sparsity in transformer activations. It has been observed that ReLU-based transformers exhibit a

high degree of activation sparsity in the feedforward network (FFN) layers (Mirzadeh et al., 2023; Sun et al., 2023c). In models like OPT-175B, over 95% of FFN activations are zero. While modern transformers favor soft activations such as GeLU and SwiGLU, replacing them with ReLU and fine-tuning has been shown to increase activation sparsity while maintaining performance (Mirzadeh et al., 2024).

Another effective sparsity technique involves weight pruning. Static weight pruning approaches, such as Wanda (Sun et al., 2023b), analyze weight importance and remove redundant connections before inference. These methods ensure a fixed sparse structure, unlike dynamic sparsity techniques that adapt weight selection based on input data. LoRA (Hu et al., 2021d) further reduces computational costs by introducing low-rank updates instead of modifying the full weight matrices. LoRA injects trainable low-rank matrices ΔW into the model:

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

With $r \ll d$, LoRA achieves significant efficiency improvements, making fine-tuning large models more memory-efficient.

3.4 Multi-Modal Fusion

Multi-modal fusion integrates information from different modality, such as text, images, and audio, to create a richer and more comprehensive representation. By capturing relationships between modalities, fusion methods enhance model understanding and response quality. Key techniques include unified feature representations, cross-modal attention mechanisms, and efficient compression strategies. These approaches enable a combined interaction between modalities while maintaining efficiency in computation and memory usage.

3.4.1 Multi-Modal Fusion Representations

Multi-Modal Key and Value Structures Unified KV representations reduce redundancy while capturing cross-modal dependencies. For a multi-modal input $x = \{x_t, x_i, x_a\}$ (text, image, and audio, respectively), the KV cache can be structured as:

$$K = \begin{bmatrix} W_k^{(t)} h^{(t)} & \alpha_{ti} W_k^{(i)} h^{(i)} & \alpha_{ta} W_k^{(a)} h^{(a)} \\ \alpha_{it} W_k^{(t)} h^{(t)} & W_k^{(i)} h^{(i)} & \alpha_{ia} W_k^{(a)} h^{(a)} \\ \alpha_{at} W_k^{(t)} h^{(t)} & \alpha_{ai} W_k^{(i)} h^{(i)} & W_k^{(a)} h^{(a)} \end{bmatrix},$$

where $W_k^{(m)}$ and $h^{(m)}$ represent the projection matrix and hidden state for modality m. The coefficients α_{mn} denote learned alignment weights between modalities m and n, enabling efficient cross-modal interaction (Jia et al., 2021; Alayrac et al., 2022; Geng et al., 2021).

Joint Cross-Modal Attention-Based Compression Given the computational overhead of storing full KV caches, cross-modal attention matrices A can be used to guide compression. For text T and image I, the attention matrix is:

$$A_{TI} = \text{diag}(K_T^\top Q_I) + \text{diag}(K_I^\top Q_T),$$

where Q and K represent the query and key matrices, respectively. A low-rank approximation of A_{TI} via Singular Value Decomposition (SVD) yields:

$$A_{TI} \approx U_{TI} \Sigma_{TI} V_{TI}^\top, \quad \Sigma_{TI} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r),$$

where $U_{TI}, \Sigma_{TI}, V_{TI}$ are the top-r components. This reduced-rank representation balances efficiency and accuracy (Jaegle et al., 2021; Hu et al., 2021a).

3.4.2 Cache Eviction on Multi-Modality Fusion

Cross-Modal Sensitivity-Based Eviction Cache eviction in multi-modal systems must account for both intra-modal and cross-modal contributions. The sensitivity of the model output y to a key k_t^m from modality m is defined as:

$$S_t^{(m)} = \left\| \frac{\partial y}{\partial k_t^m} \right\|_2^2 + \sum_{n \neq m} \beta_{mn} \left\| \frac{\partial y}{\partial k_t^n} \cdot \frac{\partial k_t^n}{\partial k_t^m} \right\|_2^2,$$

where β_{mn} captures the dependency strength between modalities m and n. Entries with the smallest $S_t^{(m)}$ are evicted, ensuring minimal performance degradation (Li et al., 2022b; Chen et al., 2022b).

Hessian-Aware Cross-Modal Eviction Second-order optimization methods are useful for assessing the importance of cache entries. The Hessian-aware importance of a key k_t^m is computed as:

$$\Delta \mathcal{L} = \frac{1}{2} \left(\frac{\partial \mathcal{L}}{\partial k_t^m}^\top H_t^{(m)} \frac{\partial \mathcal{L}}{\partial k_t^m} + \sum_{n \neq m} \frac{\partial \mathcal{L}}{\partial k_t^m}^\top H_t^{(mn)} \frac{\partial \mathcal{L}}{\partial k_t^n} \right),$$

where $H_t^{(m)}$ is the Hessian for modality m, and $H_t^{(mn)}$ captures cross-modal dependencies. Cache entries with minimal $\Delta \mathcal{L}$ are prioritized for eviction, preserving critical cross-modal interactions (Alayrac et al., 2022; Zhang et al., 2022).

Entropy-Regularized Cross-Modal Eviction Maintaining diversity in cache content can be achieved through entropy-based eviction. The joint entropy of a key distribution $p_t^{(m,n)}$ across modalities m and n is:

$$H_t = - \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{M}} p_t^{(m,n)} \log p_t^{(m,n)},$$

where $p_t^{(m,n)}$ is the normalized attention contribution of k_t for modalities m and n. Entries with low H_t are evicted, promoting diversity and informativeness (Wang et al., 2022; Kumar et al., 2022).

3.5 Efficient Sequence Modeling

Transformer-based LLMs has achieved state-of-the-art results across a variety of tasks Vaswani et al. (2017); Achiam et al. (2023); Touvron et al. (2023b). The self-attention of Transformers will achieve an all-to-all information routing between tokens in a sequence, which will result in a long sequence dependent representations Bahdanau et al. (2014). However, self-attention scales quadratically with sequence length, leading to high computational costs during training and inference. Besides, The key-value caching during autoregressive generation further increases memory overhead Tay et al. (2022).

To mitigate these efficiency challenges, several alternative architectures have been proposed, including **linear attention**, **state-space models**, **long convolution**, and **hybrid methods**, each offering distinct computational advantages over standard transformers.

Linear Attention Linear attention restructures the self-attention computation to avoid the explicit quadratic complexity of softmax attention. Given standard attention:

$$A = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}} \right) V,$$

where $Q, K, V \in \mathbb{R}^{N \times d}$ represent query, key, and value matrices, linear attention instead decomposes the softmax function using kernel-based approximations:

$$O = \phi(Q) \left(\phi(K)^\top V \right),$$

where $\phi(\cdot)$ is a feature transformation designed to preserve the key properties of softmax while enabling efficient factorization Katharopoulos et al. (2020); Choromanski et al. (2021); Qin et al. (2022). Variants of $\phi(x)$ include: - $\phi(x) = 1 + \text{elu}(x)$ Katharopoulos et al. (2020). - Cosine approximations Qin et al. (2022). - Random feature-based mappings Choromanski et al. (2021); Zhang et al. (2023a).

Despite achieving $\mathcal{O}(Nd^2)$ complexity, causal implementations require cumulative summation (`cumsum`) operations, limiting their efficiency in autoregressive settings Hua et al. (2022). Lightning Attention Qin et al. (2024a;b) resolves this through a structured block-wise decomposition that eliminates `cumsum`, maintaining efficient scaling across sequence lengths.

State Space Models State-space models (SSMs) frame sequence modeling as a continuous dynamical system:

$$h'(t) = Ah(t) + Bx(t),$$
$$y(t) = Ch(t),$$

where A, B, C are learned transition matrices controlling the evolution of hidden states Gu et al. (2022a). Unlike self-attention, SSMs process sequences with constant complexity, making them well-suited for long-context modeling Gu et al. (2022b;c).

Mamba Gu and Dao (2024) improves upon standard SSMs by introducing *selective state updates*, dynamically adjusting recurrence parameters based on input tokens. The discrete recurrence formulation is given by:

$$h_t = Ah_{t-1} + Bx_t, \quad y_t = Ch_t.$$

This design eliminates the need for extensive key-value caching, ensuring efficient autoregressive inference with linear complexity.

Long Convolution Long convolutional models replace explicit attention mechanisms with large-kernel convolutions that capture long-range dependencies Qin et al. (2023); Fu et al. (2023). The core operation for a sequence X with kernel W is:

$$O_t = \sum_{i=0}^{L} W_i X_{t-i}.$$

To accelerate training, Fast Fourier Transforms (FFT) reduce computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. However, causal convolutions require caching all prior activations, leading to high memory usage in autoregressive applications.

Hybrid Models: Combining Attention and SSMs While alternative architectures provide computational benefits, self-attention remains superior in capturing in-context relationships and performing few-shot generalization. Hybrid models, such as Mamba-2-Hybrid Waleffe et al. (2024), integrate selective self-attention with state-space models, achieving a balance between efficiency and model expressivity. Empirical studies show that hybrid architectures outperform both transformers and pure SSMs in long-context tasks, efficiently handling sequences up to 128K tokens.

Recent advances in sequence modeling reduce computational overhead. Linear attention employs kernel approximations to optimize efficiency, state-space models leverage structured recurrence for constant-time token generation, and long convolutional introduces FFT-based acceleration, etc. These efficient sequence processing techniques help reduce memory usage and speed up computation while keeping the model's ability to understand long sequences. They make it possible to handle extremely long contexts in large-scale language models.

4 Model System Co-Design for Efficient Foundation Models

4.1 Mixture of Experts (MoE)

The concept of the Mixture of experts, first introduced in (Jacobs et al., 1991; Jordan and Jacobs, 1994) has been thoroughly studied and developed through numerous subsequent works (Collobert et al., 2001;

Model Type	Time Complexity	Space Complexity
Softmax Attention	$\mathcal{O}(N^2d)$	$\mathcal{O}(N^2)$
Linear Attention	$\mathcal{O}(Nd^2)$	$\mathcal{O}(Nd)$
State-Space Models (SSMs)	$\mathcal{O}(Nd)$	$\mathcal{O}(Nd)$
Hybrid (SSM + Attention)	$\mathcal{O}(Nd + Sd^2)$	$\mathcal{O}(Nd)$
Multi-Query Attention (MQA)	$\mathcal{O}(Nd^2)$	$\mathcal{O}(Nd)$
Grouped-Query Attention (GQA)	$\mathcal{O}(Nd^2)$	$\mathcal{O}(Nd)$
Multi-Head Attention (MHA)	$\mathcal{O}(N^2d)$	$\mathcal{O}(N^2)$
Multi-Head Latent Attention (MLA)	$\mathcal{O}(Ndr)$	$\mathcal{O}(Nr)$

Table 1: Comparison of sequence modeling approaches. N: sequence length, d: feature dimension, S: state dimension, r: reduced latent dimension in MLA.

Rasmussen and Ghahramani, 2001; Eigen et al., 2014; Theis and Bethge, 2015; Aljundi et al., 2017). The advent of sparsely-gated MoE (Shazeer et al., 2017), especially in combination with transformer-based large language models (Lepikhin et al., 2020a), has revitalized this technology, which has been evolving for over three decades. The MoE framework operates on a straightforward but impactful principle: distinct components of the model, referred to as experts, are designed to specialize in specific tasks or data aspects. In this approach, only the relevant experts are activated for a given input, balancing computational efficiency with access to a vast pool of specialized expertise. This scalable and adaptable framework provides an effective solution for adhering to the scaling law, enabling increased model capacity without a proportional rise in computational costs.

The mixture of experts (MoE) approach has continued to experience significant growth, with notable advancements in 2024, including the introduction of Mixtral-8x7B (Jiang et al., 2024) and several other large-scale industrial language models such as Grok-1, DBRX, Arctic, DeepSeek-V2 (DeepSeek-AI and Liu, 2024a), and DeepSeek-V3 (DeepSeek-AI and Liu, 2024b) among others. In this work, we will use the taxonomy introduced in (Cai et al., 2024). Under this taxonomy, all the works on MoE are associated with one of three aspects: algorithmic design, system design, and application.

4.1.1 Algorithmic Design for MoE

The Mixture of Experts (MoE) framework utilizes various gating functions to determine which experts to activate for processing input data. These gating functions can be categorized into three types: sparse, dense, and soft gating.

Dense Gating Dense gating activates all experts during each iteration. The output of the dense MoE layer can be formulated as:

$$y = \sum_{i=1}^{n} G(x)_i . E_i(x)$$

The y is the output of MoE, G is the router output for the i-th expert, and E is the output of the i-th expert. The G or gating function is as below:

$$G_\sigma(x) = Softmax(x.W_g)$$

Sparse Gating In contrast, this method activates a selected subset of experts for each input token, allowing for conditional computation. The gating function can be expressed as:

$$G_\sigma(x) = Softmax(KeepTopK(H(x), K))$$

$$H(x)_i = (x.W_g)_i$$

$$KeepTopK(v, k)_i = \begin{cases} v_i & v_i \text{ in top k elements of } v \\ -\infty & otherwise \end{cases}$$

While the sparse gate $G_\sigma(x)$ significantly increases the model's parameter capacity without proportionally raising computational costs, it can introduce a load balancing problem. This issue arises when the workload is unevenly distributed among the experts, causing some to be heavily utilized while others are rarely or never activated.

Auxiliary Loss Functions To address this, each MoE layer incorporates an auxiliary loss function that promotes an even distribution of tokens across experts within each batch, as described in many studies (Lepikhin et al., 2020b; Jiang et al., 2024; Du and Huang, 2022; Lieber et al., 2024; Dai et al., 2024). To formulate this concept, consider a batch of queries $B = \{x_1, x_2, ..., x_T\}$, comprising T tokens, and N experts indexed from $i = 1$ to N . Following [28], [36], the auxiliary load balancing loss for the batch is defined as:

$$L_{load-balancing} = N \sum_{i=1}^{N} D_i P_i,$$

$$D_i = \frac{1}{T} \sum_{x \in B} 1\{argmax G_\sigma(x) = i\},$$

$$P_i = \frac{1}{T} \sum_{x \in B} G_\sigma(x)_i$$

where D_i represents the proportion of tokens distributed to expert i, while P_i denotes the proportion of the gating probability assigned to expert i. To ensure an even distribution of the batch of tokens across the N experts, the load-balancing loss function $L_{load-balancing}$ should be minimized.

Soft Gating The allocation of experts to input tokens presents a discrete optimization challenge in sparse MoE, often requiring heuristic auxiliary losses to balance expert utilization and minimize unassigned tokens. These challenges are exacerbated in out-of-distribution scenarios such as small inference batches, novel inputs, or transfer learning. Soft MoE addresses these issues by maintaining full differentiability and using all experts for processing each input, avoiding the pitfalls of discrete expert selection.

(Puigcerver et al., 2024) introduced Soft MoE, which replaces sparse, discrete gating with a soft assignment strategy that merges tokens. This approach computes weighted averages of tokens, with weights dependent on both tokens and experts, and processes each aggregate with the corresponding expert. Experimental results in image classification show that Soft MoE improves gating function training stability and maintains balance.

4.1.2 System Design for MoE

The mixture of Experts (MoE) enhances large language models but introduces challenges due to its sparse and dynamic computational workload. GShard (Lepikhin et al., 2020a) addresses this with expert parallelism, implementing parallel gating and expert computation by dispatching partitioned local tokens with load-balancing constraints. This strategy, which extends data parallelism (Rajbhandari et al., 2020; Ren et al., 2021; Rajbhandari et al., 2021), assigns each MoE expert to a distinct device while duplicating non-expert layers across devices, enabling efficient scaling of MoE models.

Computation Although MoE scales model parameters efficiently without increasing computational costs, it faces challenges in computational efficiency. A key issue is the uneven load distribution in expert parallelism, causing synchronization delays as the system waits for the most loaded expert to finish. Solutions include optimized gating mechanisms, expert capacity adjustments, and dynamic expert placement strategies introduced by SE-MoE (Shen et al., 2022), Tutel (Hwang et al., 2023), FlexMoE (Nie et al., 2023), and SmartMoE (Zhai et al., 2023), which aim to balance workloads across devices. FasterMoE (He et al., 2022) addresses severe imbalances with a dynamic shadowed expert strategy, replicating experts on multiple devices. These placement strategies influence both computation and communication efficiency.

Communication In expert parallelism, the repeated All-to-All communication during forward and backward propagation in each MoE layer creates significant overhead, often limiting efficiency. This communication spans intra-node (e.g., PCIe, NVLink) and inter-node (e.g., Ethernet, Infiniband) channels, with performance affected by bandwidth heterogeneity, network topology, and collective communication algorithms. Load imbalances in MoE further increase synchronization delays.

To address these challenges, DeepSpeed-MoE (Rajbhandari et al., 2022), HetuMoE (Nie et al., 2022), and ScheMoE (Shi et al., 2024) implement hierarchical All-to-All strategies to prioritize intra-node communication and reduce inter-node exchanges. FasterMoE (He et al., 2022), TA-MoE (Chen et al., 2022a), and SE-MoE (Shen et al., 2022) introduce topology-aware routing to minimize cross-node expert selection and inter-node communication. ExFlow (Yao et al., 2024) improves efficiency by leveraging expert affinity, keeping token processing within local GPUs. These strategies, combined with optimized placement of non-expert modules, enhance network efficiency and overall performance in distributed MoE systems (Rajbhandari et al., 2022; Singh et al., 2023; Wei et al., 2024).

Storage The growing parameter sizes in MoE models strain memory capacity, a challenge already significant in dense models. Expert parallelism helps distribute experts across devices, but individual devices, especially edge devices (e.g., PCs, smartphones, IoTs), may still face storage limitations during inference.

Solutions like SE-MoE (Shen et al., 2022), Pre-gated MoE (Hwang et al., 2024), and EdgeMoE (Yi et al., 2023) address this by retaining only essential non-expert and active expert parameters in GPU High-Bandwidth Memory (HBM), while offloading inactive parameters to CPU memory or SSDs. To reduce overhead from data transfers, these methods incorporate expert selection forecasting and prefetching to overlap parameter access with computation.

4.1.3 Application for MoE

Natural Language Processing The integration of MoE architectures into large language models (LLMs) has significantly advanced natural language understanding (NLU) and generation (NLG) tasks, including machine translation (Shazeer et al., 2017; Team and Costa-jussà, 2022), open-domain question answering (Du and Huang, 2022; Artetxe and Bhosale, 2022), code generation (Jiang et al., 2024; Wei et al., 2024; Dai et al., 2024), and mathematical problem solving (Jiang et al., 2024; DeepSeek-AI and Liu, 2024a; Dai et al., 2024). Details on incorporating MoE into LLMs are covered in algorithm design and system design. Additionally, MoE has improved LLM safety without compromising usability, as demonstrated by MoGU (Du et al., 2024), which employs dynamic routing to balance contributions between usable and safe LLMs.

Computer Vision The success of sparsely-gated Mixture of Experts (MoE) in NLP has inspired its use in computer vision. (Riquelme et al., 2021) proposed Vision MoE (V-MoE), integrating sparsely activated MLPs into select ViT blocks. In image recognition, V-MoE matches state-of-the-art performance while reducing inference computational costs, showcasing MoE's ability to capture distinct image semantics through specialized experts. Additionally, (Hwang et al., 2023) introduced Tutel, a scalable MoE system with dynamic parallelism and pipelining, demonstrated using SwinV2-MoE, built on Swin Transformer V2 (Liu et al., 2021b).

Multimodal Application Multimodal models process and integrate multiple data types, often combining images and text (Baltrušaitis et al., 2019). Mixture of Experts (MoE) architectures are foundational for these models due to their ability to specialize expert layers for different modalities. A key example is LIMoE (Mustafa et al., 2022), a sparse MoE model for multimodal learning trained with contrastive loss and entropy-based regularization to address load balancing. Further advancements by (Shen et al., 2023a; Li et al., 2023b; Lin et al., 2024) have enhanced the scaling of vision-language models.

To tackle task conflicts in instruction tuning of Large Vision-Language Models (LVLMs), MoCLE (Gou et al., 2024) combines MoE with LoRA (Hu et al., 2021e) experts and a universal expert for task-specific parameter activation. Similarly, LLaVAMoLE (Chen et al., 2024b) mitigates data conflicts in Multimodal Large Language Models (MLLMs) using LoRA experts for the MLP layer and a top-1 gating mechanism to refine instruction tuning.

4.2 Mixed Precision Training (MPT)

Mixed precision training integrates multiple numerical formats, such as FP32, FP16, FP8 and INT8, to optimize computational efficiency for large-scale deep learning models. By selectively assigning precision to various components of the model, it reduces memory usage and accelerates computation while preserving accuracy. A critical aspect of this approach is the dynamic assignment of precision, where components that significantly impact model performance are kept in high precision, while less critical ones are quantized. For example, let $\mathcal{F}(X;W)$ represent the output of a neural network layer with input X and weights W. In a mixed precision setting, the weights can be decomposed as $W = W_{\text{high}} + Q_{\text{low}}$, where W_{high} remains in FP32, and Q_{low} is a quantized representation in FP16 or INT8. The optimal precision assignment $\mathcal{P}$ minimizes the reconstruction error $\|\mathcal{F}(X;W) - \mathcal{F}(X;\mathcal{P}(W))\|_2^2$, subject to $\mathcal{P}(W) \in \{\text{FP32}, \text{FP16}, \text{FP8}, \text{INT8}, etc\}$ (Zhang et al., 2023d; Micikevicius et al., 2017; Zhao et al., 2021; Piao et al., 2022).

A notable application of precision assignment arises in attention mechanisms, where the key-value cache, denoted by $K \in \mathbb{R}^{d_k \times n}$ and $V \in \mathbb{R}^{d_v \times n}$, can benefit from dynamic scaling. For each key vector k_j, its precision can be adjusted based on an importance score. Specifically, the precision map $\mathcal{P}(k_j)$ is designed to minimize the combined reconstruction error and alignment loss (Smith et al., 2024). This optimization ensures that high-magnitude vectors remain in FP32, while low-magnitude ones are quantized for efficiency.

$$\min_{\mathcal{P}} \mathbb{E}_X \left[\sum_{j=1}^{n} \left(\|k_j - \mathcal{P}(k_j)\|_2^2 + \lambda \cdot \|\text{softmax}(K^\top X) - \text{softmax}(\mathcal{P}(K)^\top X)\|_F^2 \right) \right].$$

4.2.1 Computation for MPT

In mixed precision training, forward and backward passes are carefully designed to maximize efficiency without compromising stability. During the forward pass, the input to each layer, $x^{(l)}$, is multiplied by the weight matrix, $W^{(l)}$, to compute the pre-activation output $z^{(l)} = W^{(l)}x^{(l)} + b^{(l)}$, where $b^{(l)}$ is the bias term. To leverage hardware acceleration, $W^{(l)}$ is stored in FP16, while $z^{(l)}$ may temporarily reside in FP32 to prevent numerical overflow during activation. For example, when ReLU is applied, $y^{(l)} = \max(0, z^{(l)})$ is computed in FP32 to maintain accuracy. In the backward pass, gradients are computed using the chain rule, where $\nabla W^{(l)} = \delta^{(l)}(x^{(l)})^\top$, and $\delta^{(l)} = \sigma'(z^{(l)}) \cdot \delta^{(l+1)}$ represents the backpropagated gradient through the activation σ. To ensure numerical stability, all gradients are accumulated in FP32, even if intermediate values are stored in FP16.

To handle mixed precision more robustly, recent works propose incorporating second-order approximations for more precise weight updates (Micikevicius et al., 2018a). For a general weight matrix W undergoing gradient descent with mixed precision, the update rule can be expressed as:

$$W_{\text{new}} = W - \eta \left[\nabla \mathcal{L}(W) + \frac{\partial^2 \mathcal{L}}{\partial W^2} \cdot Q_{\text{low}} \right],$$

where $\frac{\partial^2 \mathcal{L}}{\partial W^2}$ is the Hessian, and Q_{low} is the quantization error. This formulation allows for dynamically correcting the impact of quantized components during training Gholami et al. (2022); Rokh et al. (2022).

4.2.2 Error Correction for MPT

Mixed Precision Training introduces errors that propagate through the layers of a neural network. Let $\mathcal{N}(X;W)$ represent the output of a neural operator with input X and weights W, and $\hat{\mathcal{N}}(X;\hat{W})$ its mixed precision approximation. Using a Taylor expansion, the error propagation can be analyzed as:

$$\mathcal{N}(X;W) - \hat{\mathcal{N}}(X;\hat{W}) \approx J_{\mathcal{N}}(W) \cdot (W - \hat{W}) + \frac{1}{2}(W - \hat{W})^\top H_{\mathcal{N}}(W)(W - \hat{W}),$$

where $J_{\mathcal{N}}(W)$ and $H_{\mathcal{N}}(W)$ denote the Jacobian and Hessian of $\mathcal{N}$, respectively. The first-order term dominates when quantization errors are small, while the second-order term becomes significant for larger errors. To minimize error propagation, recent studies introduce regularization terms in the loss function (Zhang

et al., 2023d):

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \beta \cdot \|J_{\mathcal{N}}(W) \cdot Q_{\text{low}}\|_F^2 + \gamma \cdot \|H_{\mathcal{N}}(W) \cdot Q_{\text{low}}\|_F^2,$$

where β and γ are hyperparameters controlling the weight of first-order and second-order regularization.

4.2.3 Theoretical Analysis for MPT

A key theoretical challenge in mixed precision training is quantifying the error introduced by low-precision computations. The approximation error ϵ can be bounded as:

$$\epsilon = \|\mathcal{N}(X; W) - \hat{\mathcal{N}}(X; \hat{W})\|_F \leq \|J_{\mathcal{N}}(W)\|_F \cdot \|W - \hat{W}\|_F + \frac{1}{2}\|H_{\mathcal{N}}(W)\|_F \cdot \|W - \hat{W}\|_F^2.$$

This bound ensures that the total error is a function of both the quantization level and the sensitivity of the network to weight perturbations, represented by the Jacobian and Hessian norms. These guarantees are critical for designing robust precision allocation strategies and ensuring reliable model performance under mixed precision training (Lee et al., 2018; Zhang et al., 2023d).

4.3 Efficient Pretraining

Efficient pretraining combines model-level innovations and system-level optimizations to enable large language models (LLMs) to scale efficiently while minimizing computational and memory costs. Recent advancements focus on optimizing transformer architectures, pretraining objectives, and data strategies while aligning these improvements with hardware capabilities, such as quantization and distributed systems. This model-system co-design approach ensures that both algorithmic performance and resource utilization are optimized.

4.3.1 Efficient Pretraining Optimization

Optimizing the transformer architecture is central to efficient pretraining. GQKVA introduces grouped attention, where queries Q, keys K, and values V are clustered to reduce redundancy in self-attention operations (Javadi et al., 2023). Let $Q, K, V \in \mathbb{R}^{n \times d_k}$ be the query, key, and value matrices. The grouped matrices $\tilde{Q}, \tilde{K}, \tilde{V} \in \mathbb{R}^{\kappa \times d_k}$ are defined by clustering with κ groups, reducing the attention complexity from $O(n^2 d_k)$ to $O(\kappa^2 d_k)$. This optimization aligns with hardware acceleration, such as tensor cores, which are efficient for smaller matrix operations.

Jetfire enhances efficiency with block-wise INT8 quantization (Xi et al., 2024). For weights $W_b \in \mathbb{R}^{d_b \times d_b}$, quantization scales each block using a factor $\Delta_b = \frac{\max(W_b) - \min(W_b)}{2^8 - 1}$, reducing memory overhead. The quantized weights are defined as:$\hat{W}_b = \text{round}\left(\frac{W_b}{\Delta_b}\right) \cdot \Delta_b$ and $W_b \approx \hat{W}_b + \epsilon$, where ϵ represents the quantization error. These techniques reduce both memory and computation costs, enabling efficient large-scale pretraining.

4.3.2 Efficient Pretraining on Distributed Systems

Continual pretraining enables general-purpose LLMs to adapt efficiently to domain-specific tasks. METRO introduces a denoising objective combined with model-generated auxiliary signals to improve robustness (Bajaj et al., 2022). Let X_m denote a masked input sequence, and let $\mathcal{F}_\theta$ be the model's prediction function. The pretraining objective is:

$$\mathcal{L}_{\text{pretrain}} = \sum_{t=1}^{n} \|X_{t,\text{true}} - \mathcal{F}_\theta(X_{m,t})\|^2 + \alpha \sum_{t=1}^{n} \|S_t - \mathcal{F}_\theta(X_{m,t})\|^2,$$

where S_t are auxiliary signals generated by the model, and α balances the two components. This objective can be efficiently distributed across GPUs using data partitioning.

FinGPT-HPC utilizes bucketed data shuffling to optimize I/O performance during distributed training (Liu et al., 2024b). Each bucket B_i is assigned to a GPU, where the time per training step is modeled as: $T_{\text{pretrain}} = \max\left(\frac{\mathcal{C}_{\text{comm}}}{N_{\text{GPUs}}}, \mathcal{C}_{\text{comp}}\right)$,

where $\mathcal{C}_{\text{comm}}$ and $\mathcal{C}_{\text{comp}}$ represent communication and computation costs. By overlapping communication and computation, this approach ensures scalability.

4.3.3 Efficient Pretraining via Sampling

Efficient pretraining also relies on prioritizing high-relevance data through task-specific sampling. Bucket Pre-training assigns each bucket B_i a relevance weight w_i, where the sampling probability is: $p_i = \frac{w_i}{\sum_{j=1}^{m} w_j}$. This ensures that critical data receives more updates, reducing overfitting to irrelevant tasks (Liu et al., 2024a).

MixMAE combines mixed and masked token reconstruction objectives to enhance the diversity of token representations (Liu et al., 2023a). The total loss is:

$$\mathcal{L}_{\text{MixMAE}} = \beta \cdot \mathcal{L}_{\text{mix}} + (1 - \beta) \cdot \mathcal{L}_{\text{mask}} + \gamma \cdot \mathcal{L}_{\text{contrast}},$$

where $\beta, \gamma \in [0, 1]$ control the weight of mixed, masked, and contrastive objectives, respectively. This hybrid strategy strengthens token representations for downstream tasks.

4.4 Efficient Fine-tuning

Fine-tuning LLMs requires balancing computational efficiency with task-specific performance. This challenge is best addressed through a co-design approach, integrating model-level techniques (e.g., parameter-efficient updates, sparsity) with system-level optimizations (e.g., memory compression, resource allocation). By jointly optimizing these layers, fine-tuning methods can achieve scalability and performance under constrained resources.

4.4.1 Efficient Fine-Tuning for Parameters

Parameter-efficient fine-tuning (PEFT) techniques, such as Low-Rank Adaptation (LoRA) and Prefix-Tuning, aim to reduce the number of trainable parameters while maintaining model performance. These methods modify the training process to focus on smaller, task-specific updates instead of adjusting all model parameters (Liu et al., 2024c; Zhang et al., 2023c; Liu et al., 2021a; Xu et al., 2023; Zhang et al., 2023b; Ding et al., 2023).

LoRA reduces memory and computation requirements by introducing a low-rank decomposition for weight updates (Liu et al., 2024c; Zhang et al., 2023c),. The model weights $W \in \mathbb{R}^{d \times d}$ are kept frozen, while updates are parameterized as $\Delta W = \mathbf{A}\mathbf{B}^\top$, where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d \times r}$ and $r \ll d$. Training only $\mathbf{A}$ and $\mathbf{B}$ minimizes resource usage while preserving the model's capacity to adapt to new tasks. To further optimize resource utilization, system-level techniques such as quantization are applied to $\mathbf{A}$ and $\mathbf{B}$. Quantization scales the matrices using step sizes (e.g., $\Delta_{\mathbf{A}}$ and $\Delta_{\mathbf{B}}$), enabling hardware-friendly computation with reduced memory overhead (Lv et al.).

Prefix-Tuning adapts the model by learning a continuous prefix embedding $P \in \mathbb{R}^{k \times d}$, which is prepended to the input embeddings. The pre-trained model weights remain unchanged. The optimization objective for Prefix-Tuning is:

$$\min_{P} \mathcal{L}_{\text{task}}(f_\theta([P; X]), Y),$$

where X is the input, Y is the target, and f_θ is the frozen pre-trained model. This method allows efficient adaptation to a wide range of tasks with minimal computational costs (Li and Liang, 2021; Xu et al., 2023; Zhang et al., 2023b; Ding et al., 2023).

System-level optimizations can further enhance the efficiency of Prefix-Tuning. For example, the prefix matrix P can be compressed using a low-rank approximation, where $P \approx \mathbf{U}_P \mathbf{V}_P^\top$ and $\mathbf{U}_P, \mathbf{V}_P \in \mathbb{R}^{k \times r}$. This reduces memory usage by storing only the decomposed components. The optimization objective then becomes:

$$\min_{\mathbf{U}_P, \mathbf{V}_P} \|\mathcal{F}_\theta([P; X]) - \mathcal{F}_\theta([\mathbf{U}_P \mathbf{V}_P^\top; X])\|_F^2 + \lambda \cdot \mathcal{C}(\mathbf{U}_P, \mathbf{V}_P),$$

where $\mathcal{C}(\mathbf{U}_P, \mathbf{V}_P)$ represents the memory cost of the decomposed components. This coupling of model compression and system-aware design ensures that Prefix-Tuning remains efficient across diverse hardware platforms (Liu et al., 2021a; Lv et al.).

By integrating low-rank decompositions and system-level optimizations, PEFT techniques demonstrate the principles of model-system co-design. These methods enable scalable fine-tuning for large models while ensuring efficient resource utilization. LoRA and Prefix-Tuning have proven effective in adapting large language models to specific tasks with minimal memory and computational costs (Liu et al., 2024c; Zhang et al., 2023c; Li and Liang, 2021; Liu et al., 2021a).

4.4.2 Efficient Fine-Tuning for Sparsity

Sparse fine-tuning leverages inherent sparsity in gradients and parameter updates to optimize computational and memory efficiency, particularly in the context of large-scale language models. Unlike traditional fine-tuning approaches that involve dense updates across all model parameters, sparse fine-tuning identifies and updates only the most impactful weights or gradients, significantly reducing resource consumption while preserving task performance.

Scaling Sparse Fine-Tuning (SSF) dynamically applies sparsity to the gradient matrix $G \in \mathbb{R}^{d\times d}$ by introducing a binary mask $M \in \{0,1\}^{d\times d}$, which selects critical elements based on a combination of gradient magnitude and computational cost. Specifically, the mask M is defined such that $M_{ij} = 1$ if $|G_{ij}|/\text{Cost}(G_{ij}) \geq \tau$, where τ is an adaptive threshold adjusted during training. The resulting sparse update is then computed as $\Delta W = M \odot G$, where $\odot$ denotes element-wise multiplication (Ansell et al., 2024). By incorporating cost metrics like memory bandwidth or latency, this method aligns gradient sparsity with hardware constraints.

Layerwise Importance Sampling for Memory-Efficient Fine-Tuning (LISA) further optimizes sparse fine-tuning by focusing on the most critical layers of the model. Each layer's importance is evaluated using the gradient norm or its contribution to the task-specific loss, $\mathcal{L}_{\text{task}}$. Formally, the layer subset $\mathcal{S}$ to be updated is selected as:

$$\mathcal{S} = \underset{\mathcal{S}\subseteq\{1,\ldots,L\}}{\arg\max} \sum_{l\in\mathcal{S}} \frac{\|\nabla_{\theta^{(l)}}\mathcal{L}_{\text{task}}\|_F}{\text{Cost}(\nabla_{\theta^{(l)}})}, \quad |\mathcal{S}| \leq k,$$

where $\nabla_{\theta^{(l)}}\mathcal{L}_{\text{task}}$ denotes the gradient of the loss with respect to layer l's parameters, and $\text{Cost}(\cdot)$ represents the resource overhead of updating that layer. This approach ensures that updates are concentrated on the most impactful layers while adhering to predefined resource budgets (Pan et al., 2024).

In addition to layerwise optimization, sparse fine-tuning benefits from system-level enhancements like gradient caching and asynchronous execution. For example, momentum-based gradient sparsity propagates prior updates to stabilize training under sparsity constraints. Let $\Delta W^{(t-1)}$ be the previous sparse update. At step t, the new sparse update integrates past momentum as:

$$\Delta W^{(t)} = M^{(t)} \odot \left(G^{(t)} + \beta \cdot \Delta W^{(t-1)}\right),$$

where β is a momentum hyperparameter that smooths the gradient trajectory across sparse updates. This formulation ensures consistent parameter updates despite high sparsity (Dettmers and Zettlemoyer, 2019; Shoeybi et al., 2019a).

Furthermore, system optimizations such as asynchronous execution enhance scalability in distributed environments. Sparse gradient updates are computed independently across N_{devices} processing units, with synchronization occurring intermittently. If T_{comp} is the computation time for sparse updates and T_{sync} the synchronization overhead, the total time per step is: $T_{\text{total}} = \max\left(T_{\text{comp}}, \frac{T_{\text{sync}}}{N_{\text{devices}}}\right)$, where overlapping computation and communication minimizes total latency (Shen et al., 2023b; Huang et al., 2019).

Sparse fine-tuning techniques have demonstrated broad applicability, ranging from domain-specific tasks (e.g., legal and biomedical text generation) to resource-constrained environments, such as edge devices or distributed training. Future directions may explore tighter integration between sparse fine-tuning and hardware-level optimizations, including tensor sparsity on GPUs and custom accelerators tailored for sparse computation (Ansell et al., 2024; Pan et al., 2024; Liu et al., 2024c; Zhang et al., 2023c).

4.5 Model-System Co-Design Unified Framework

Efficient pretraining leverages model and system optimizations to balance accuracy, memory, and computational efficiency. A unified framework integrates key aspects of quantization, sparsity, and distributed system design to maximize efficiency. The objective is to optimize pretraining by explicitly modeling system constraints and pretraining-specific features.

Let $\mathcal{L}_{\text{task}}(\theta)$ represent the pretraining loss, where θ includes parameters such as weights W, quantization factors Δ, and sparsity masks $\mathbf{M}$. To enhance pretraining efficiency, weights are quantized and sparse updates are applied. Quantized weights are expressed as $W = \mathcal{Q}(\hat{W}, \Delta)$, where $\mathcal{Q}(\hat{W}, \Delta)$ denotes quantization with step size Δ. Sparse masks $\mathbf{M}$ restrict updates to the most critical elements, ensuring computational efficiency.

The unified objective integrates the pretraining loss with quantization and sparsity constraints:

$$\mathcal{L}_{\text{pretrain}} = \sum_{i=1}^{L} \|\mathbf{M}_i \odot \mathcal{Q}(\hat{W}_i, \Delta_i)\|_F^2 + \sum_{j=1}^{n} \|\mathbf{M}_{A,j} \odot \mathcal{F}_\theta(X_j) - Y_j\|^2,$$

where: - $\mathbf{M}_i$ represents the weight sparsity mask for layer i. - $\mathbf{M}_{A,j}$ denotes the attention sparsity mask for token j. - $\mathcal{F}_\theta(X_j)$ is the model output for input X_j, compared to target Y_j.

5 Conclusion

Efficient Foundation Model Design techniques aim to build models that are computationally efficient while achieving faster training and inference speeds without sacrificing performance. These techniques involve three main areas: model design, system design, and model-system co-design.

Model design focuses on optimizing the internal structure of models for acceleration. Quantization provides a hardware-focused perspective, compressing models into lower-precision representations to optimize storage and computation. Knowledge distillation allows efficient knowledge transfer, enabling smaller student models to study the performance from larger teacher models. Pruning offers a neural network model reduction by eliminating redundant connections.

System design introduces optimizations at the system design infrastructure level, it include KV cache compression, parallelism, contextual sparsity, etc. KV cache compression minimizes memory usage by reducing the storage of key-value caches, which is efficient in long-context processing. Parallelism improves throughput by distributing computation across multiple GPUs/CPUs. Sparsity eliminates unnecessary computations in sparse attention matrix, which further reduces overhead. Low-level GPU optimizations maximize hardware utilization, which can achieve faster and more scalable model execution.

Model-system co-design bridges the gap between model and system design. MoE enables dynamic routing to reduce computation while maintaining performance for diverse tasks. Fine-tuning techniques enhance adaptability on downstream tasks. Mixed-precision training reduces computation by leveraging both low- and high-precision computation. These model and system co-design strategies leads to better resource utilization and faster inference.

References

Joshua Achiam, Raphaël Gontijo Lopes, Sylvain Gelly, Ekin Dogus Cubuk, Simon Kornblith, and David J. Fleet. Scaling laws for neural language models. *ArXiv*, cs.LG, 2023.

Jean-Baptiste Alayrac et al. Flamingo: A visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*, 2023.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873, 2024.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M Ponti. Scaling sparse fine-tuning to large language models. *arXiv preprint arXiv:2401.16405*, 2024.

Mikel Artetxe and Shruti Bhosale. Efficient large scale language modeling with mixtures of experts, 2022. URL https://arxiv.org/abs/2112.10684.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

P Bajaj, C Xiong, G Ke, X Liu, D He, S Tiwary, TY Liu, P Bennett, X Song, and J Gao. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals (arxiv: 2204.06644). arxiv. *arXiv preprint arXiv.2204.06644*, 2022.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019. doi: 10.1109/TPAMI.2018.2798607.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. Efficient 8-bit quantization of transformer neural machine language translation model. *arXiv preprint arXiv:1906.00532*, 2019.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023. URL http://arxiv.org/abs/2210.09461.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. URL https://arxiv.org/abs/2303.12712.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204*, 2024.

Chang Chen, Min Li, Zhihua Wu, Dianhai Yu, and Chao Yang. Ta-moe: Topology-aware large scale mixture-of-expert training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22173–22186. Curran Associates, Inc., 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8b465dd58ac50e1b0b22894fd581f62f-Paper-Conference.pdf.

Chen Chen et al. Quest: Query-aware sparsity for long-context transformers. *NeurIPS 2024*, 2024a.

Rui Chen et al. Hessian-based optimization for multi-modal attention models. *ICLR*, 2022b.

Shaoxiang Chen, Zequn Jie, and Lin Ma. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms, 2024b. URL https://arxiv.org/abs/2401.16160.

Xinzhe Chen, Yuan Gao, Jinsong Zhang, and Xiaodong Zhou. Distilling knowledge via knowledge transfer in text-to-text models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 260–267, 2020. URL https://aclanthology.org/2020.acl-main.24/.

Brian Chmiel, Ron Banner, Elad Hoffer, Hilla Ben-Yaacov, and Daniel Soudry. Accurate neural training with 4-bit matrix multiplications at standard formats. In *The Eleventh International Conference on Learning Representations*, 2023.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *International Conference on Learning Representations (ICLR)*, 2021.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of svms for very large scale problems. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/36ac8e558ac7690b6f44e2cb5ef93322-Paper.pdf.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

Tri Dao et al. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022.

DeepSeek-AI and Aixin Liu. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024a. URL https://arxiv.org/abs/2405.04434.

DeepSeek-AI and Aixin Liu. Deepseek-v3 technical report, 2024b. URL https://arxiv.org/abs/2412.19437.

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. In *arXiv preprint arXiv:1907.04840*, 2019.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. *arXiv preprint arXiv:2311.11696*, 2023.

H. Dong, B. Liu, and K. Wang. Packqvit: Faster sub-8-bit vision transformers via full and packed quantization on the mobile. In *Proceedings of NeurIPS 2023*, pages 4567–4578, 2023.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. *arXiv preprint arXiv:2406.02924*, 2024.

Nan Du and Huang. GLaM: Efficient scaling of language models with mixture-of-experts. In Chaudhuri, editor, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Yanrui Du, Sendong Zhao, Danyang Zhao, Ming Ma, Yuhan Chen, Liangyu Huo, Qing Yang, Dongliang Xu, and Bing Qin. Mogu: A framework for enhancing safety of open-sourced llms while preserving their usability, 2024. URL https://arxiv.org/abs/2405.14488.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts, 2014. URL https://arxiv.org/abs/1312.4314.

Steven K. Esser, Jeffrey L. McKinstry, Devesh Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 1, 2022a.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022b.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2023. URL https://arxiv.org/abs/2210.17323.

Shangyu Fu, Xiaolong Zou, Weijia Zhang, Kaiwen Zheng, and Zhihui Li. Efficient long-range sequence modeling with fourier-transformed convolution. *ArXiv*, cs.LG, 2023.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Yijun Geng et al. M3ae: Multi-modal masked autoencoders for efficient representation learning. *arXiv preprint arXiv:2111.05234*, 2021.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

Rania Girgis, Shalitha Mathew, and Kassem Ali. Effective methods for improving interpretability in reinforcement learning models. *Artificial Intelligence Review*, 45(3):1–14, 2021. Available at: https://link.springer.com/article/10.1007/s10462-021-09984-x.

Yeyun Gong, Jinglu Chen, Changjian Wang, Han Zhang, Haoyang Liu, Yu Sun, Weizhu Chen, Bowen Zhou, and Tie-Yan Liu. Preserving knowledge in pre-trained language models via knowledge distillation. *arXiv preprint arXiv:2203.02436*, 2022. URL https://arxiv.org/abs/2203.02436.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T. Kwok, and Yu Zhang. Mixture of cluster-conditional lora experts for vision-language instruction tuning, 2024. URL https://arxiv.org/abs/2312.12379.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *ArXiv*, cs.LG, 2024.

Albert Gu, Tri Dao, Stefano Ermon, and Christopher Ré. Efficient training of long sequence models with structured state spaces. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a.

Albert Gu, Karan Goel, and Christopher Ré. State space models for sequence modeling. *International Conference on Machine Learning (ICML)*, 2022b.

Albert Gu, Karan Goel, and Christopher Ré. Efficient training of long sequence models with structured state spaces. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022c.

Naibin Gu, Peng Fu, Xiyu Liu, Bowen Shen, Zheng Lin, and Weiping Wang. Light-peft: Lightening parameter-efficient fine-tuning via early pruning. *arXiv preprint arXiv:2406.03792*, 2024a.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024b. URL https://arxiv.org/abs/2306.08543.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. Fastermoe: modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '22, page 120–134, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392044. doi: 10.1145/3503221.3508418. URL https://doi.org/10.1145/3503221.3508418.

Yefei He, Feng Chen, Jing Liu, et al. Zipvl: Efficient large vision-language models with dynamic token sparsification and kv cache compression. *arXiv preprint arXiv:2410.08584*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

Le Hou, Zhewei Yu, Fei Chen, Peng Jin, Zhenyu Yang, Yen-Kuang Cheng, and Lin Xu. Dynabert: Dynamic bert with adaptive width and depth. In *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793, 2020.

Edward Hu et al. Low-rank adaptation for cross-modal learning. *arXiv preprint arXiv:2106.09685*, 2021a.

Edward Hu et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021b.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021c.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021d. URL https://arxiv.org/abs/2106.09685.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021e. URL https://arxiv.org/abs/2106.09685.

Wen Hua, Zhen Qin, Dong Li, Weigao Sun, and Yiran Zhong. Causal linear transformers: Overcoming cumulative sum bottlenecks in autoregressive inference. *ArXiv*, cs.LG, 2022.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. *arXiv preprint arXiv:2402.04291*, 2024.

Weizhe Huang et al. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2305.06396*, 2023.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mingxing Chen, Zhenzhong Chen, Fei Tan, Yashuo Sugawara, Wei Wang, Maxim Krikun, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 103–112, 2019.

Zhecheng Huang, Wanjun Zou, Yuwei Wang, and Eric Xing. Towards understanding chain-of-thought prompting: An empirical study of multitask learning in context. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5100–5114. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.emnlp-main.359.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, HoYuen Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. Tutel: Adaptive mixture-of-experts at scale. In D. Song, M. Carbin, and T. Chen, editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 269–287. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/5616d34cf8ff73942cfd5aa922842556-Paper-mlsys2023.pdf.

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 1018–1031, 2024. doi: 10.1109/ISCA59077.2024.00078.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Andrew Jaegle et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

Vahid Jafari, Haitao Liu, and Qing Lin. Sequence-level knowledge distillation for low-resource neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 2134–2145, 2021. URL https://aclanthology.org/2021.findings-emnlp.178/.

Farnoosh Javadi, Walid Ahmed, Habib Hajimolahoseini, Foozhan Ataiefard, Mohammad Hassanpour, Saina Asani, Austin Wen, Omar Mohamed Awad, Kangling Liu, and Yang Liu. Gqkva: Efficient pre-training of transformers by grouping queries, keys, and values. *arXiv preprint arXiv:2311.03426*, 2023.

Mengyuan Jia et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Qing Jin, Jian Ren, Richard Zhuang, Sumant Hanumante, Zhengang Li, Zhiyu Chen, Yanzhi Wang, Kaiyuan Yang, and Sergey Tulyakov. F8net: Fixed-point 8-bit only multiplication for network quantization. *arXiv preprint arXiv:2202.05239*, 2022.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Jongwoo Ko, Seungjoon Park, Yujin Kim, Sumyeong Ahn, Du-Seong Chang, Euijai Ahn, and Se-Young Yun. Nash: A simple unified framework of structured pruning for accelerating encoder-decoder language models. *arXiv preprint arXiv:2310.10054*, 2023.

Ramesh Kumar et al. Cross-modal fusion strategies for efficient model inference. *arXiv preprint arXiv:2205.12345*, 2022.

Mark Kurtz et al. Inducing contextual sparsity in transformers for faster inference. *arXiv preprint arXiv:2005.14187*, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Dong-Ik Lee et al. Mixed precision training of convolutional neural networks using integer operations. *arXiv preprint arXiv:1802.00930*, 2018.

Dongyoung Lee, Seungkyu Choi, and Ik Joon Chang. Qrazor: Reliable and effortless 4-bit llm quantization by significant data razoring. *arXiv preprint arXiv:2501.13331*, 2025. URL https://arxiv.org/abs/2501.13331.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020a. URL https://arxiv.org/abs/2006.16668.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020b. URL https://arxiv.org/abs/2006.16668.

Jinhao Li, Jiaming Xu13, Shiyao Li23, Shan Huang, Jun Liu, Yaoxiu Lian, and Guohao Dai13. Fast and efficient 2-bit llm inference on gpu: 2/4/16-bit in a weight matrix with asynchronous dequantization. 2024.

Michael Li et al. Deja vu: Improving memory access for large language models. *arXiv preprint arXiv:2207.06498*, 2022a.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Xiaodong Li et al. Sensitivity-aware pruning for efficient multi-modal transformers. *NeurIPS*, 2022b.

Xin-Chun Li, Wen-Shu Fan, Shaoming Song, Yinchuan Li, Bingshuai Li, Yunfeng Shao, and De-Chuan Zhan. Asymmetric temperature scaling makes larger networks teach well again. In *Advances in Neural Information Processing Systems*, volume 35, pages 12345–12356, 2022c.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023a. URL https://arxiv.org/abs/2302.04304.

Yunshui Li, Binyuan Hui, ZhiChao Yin, Min Yang, Fei Huang, and Yongbin Li. Pace: Unified multi-modal dialogue pre-training with progressive and compositional experts, 2023b. URL https://arxiv.org/abs/2305.14839.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024. URL https://arxiv.org/abs/2403.19887.

Bin Lin, Zhenyu Tang, Yang Ye, Jinfa Huang, Junwu Zhang, Yatian Pang, Peng Jin, Munan Ning, Jiebo Luo, and Li Yuan. Moe-llava: Mixture of experts for large vision-language models, 2024. URL https://arxiv.org/abs/2401.15947.

Hongtao Liu, Qiyao Peng, Qing Yang, Kai Liu, and Hongyan Xu. Bucket pre-training is all you need. *arXiv preprint arXiv:2407.07495*, 2024a.

Jihao Liu, Xin Huang, Jinliang Zheng, Yu Liu, and Hongsheng Li. Mixmae: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6252–6261, 2023a.

Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. Llm-fp4: 4-bit floating-point quantized transformers. *arXiv preprint arXiv:2310.16836*, 2023b.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021a.

Xiao-Yang Liu, Jie Zhang, Guoxuan Wang, Weiqing Tong, and Anwar Walid. Fingpt-hpc: Efficient pre-training and finetuning large language models for financial applications with high-performance computing. *arXiv preprint arXiv:2402.13533*, 2024b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021b.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. Alora: Allocating low-rank adaptation for fine-tuning large language models. *arXiv preprint arXiv:2403.16187*, 2024c.

K Lv, Y Yang, T Liu, Q Gao, Q Guo, and X Qiu. Full parameter fine-tuning for large language models with limited resources. arxiv 2023. *arXiv preprint arXiv:2306.09782*.

X. Ma, G. Fang, and X. Wang. Llm-pruner: On the structural pruning of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720, 2023.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Paulius Micikevicius et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2018a.

Paulius Micikevicius et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2018b.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.naacl-main.201.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Niranjan Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198, 2020. doi: 10.1609/aaai.v34i04.5948.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023. URL https://arxiv.org/abs/2310.04564.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2024. URL https://arxiv.org/abs/2310.04564.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9564–9576. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3e67e84abf900bb2c7cbd5759bfce62d-Paper-Conference.pdf.

Markus Nagel, Raoul Amjad, Mart Van Baalen, et al. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Xiaonan Nie, Pinxue Zhao, Xupeng Miao, Tong Zhao, and Bin Cui. Hetumoe: An efficient trillion-scale mixture-of-expert distributed training system, 2022. URL https://arxiv.org/abs/2203.14685.

Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. Flexmoe: Scaling large-scale sparse pre-trained model training via dynamic device placement. *Proc. ACM Manag. Data*, 1(1), May 2023. doi: 10.1145/3588964. URL https://doi.org/10.1145/3588964.

NVIDIA. Fastertransformer: An efficient transformer library. https://github.com/NVIDIA/FasterTransformer, 2021. Accessed: 2023-10-08.

NVIDIA. Tensorrt: A high performance deep learning inference library. https://github.com/NVIDIA/TensorRT, 2022. Accessed: 2023-10-08.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*, 2024.

Jian Peng et al. Cachegen: Kv cache compression and streaming for fast large language model serving. *arXiv preprint arXiv:2211.07935*, 2022.

Tairen Piao, Ikhyun Cho, and U Kang. Sensimix: Sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for bert compression. *PloS one*, 17(4):e0265621, 2022.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jxpsAj7ltE.

Zhen Qin, Dong Li, Weigao Sun, et al. Linear transformer with gated recurrent network for long sequence modeling. *ArXiv*, cs.LG, 2022.

Zhen Qin, Weigao Sun, Dong Li, and Yiran Zhong. Long convolutional sequence models: Extending the receptive field of convolutions for efficient sequence processing. *ArXiv*, cs.LG, 2023.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention: Linear attention with constant training speed. *ArXiv*, cs.CL, 2024a.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *ArXiv*, cs.CL, 2024b.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2020. doi: 10.1109/SC41405.2020.00024.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 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*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476205. URL https://doi.org/10.1145/3458817.3476205.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.htm.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021.

Carl Rasmussen and Zoubin Ghahramani. Infinite mixtures of gaussian process experts. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/9afefc52942cb83c7c1f14b2139b09ba-Paper.pdf.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. ZeRO-Offload: Democratizing Billion-Scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564. USENIX Association, July 2021. ISBN 978-1-939133-23-6. URL https://www.usenix.org/conference/atc21/presentation/ren-jie.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8583–8595. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. A comprehensive survey on model quantization for deep neural networks. *arXiv preprint arXiv:2205.07877*, 2022.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Maksym Shamrai. Language-specific pruning for efficient reduction of large language models. In *Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP)@ LREC-COLING 2024*, pages 135–140, 2024.

Hang Shao, Bei Liu, and Yanmin Qian. One-shot sensitivity-aware mixed sparsity pruning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11296–11300. IEEE, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Liang Shen, Zhihua Wu, Weibao Gong, Hongxiang Hao, Yangfan Bai, HuaChao Wu, Xinxuan Wu, Haoyi Xiong, Dianhai Yu, and Yanjun Ma. Se-moe: A scalable and efficient mixture-of-experts distributed training and inference system. *CoRR*, abs/2205.10034, 2022. URL https://doi.org/10.48550/arXiv.2205.10034.

Sheng Shen, Zhewei Yao, Chunyuan Li, Trevor Darrell, Kurt Keutzer, and Yuxiong He. Scaling vision-language models with sparse mixture of experts, 2023a. URL https://arxiv.org/abs/2303.07226.

Wei Shen et al. Flexgen: High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2302.07800*, 2023b.

Y. Shen, X. Zhang, and J. Lin. Efficient post-training quantization with fp8 formats. *NeurIPS*, 37:1234–1245, 2024.

Shaohuai Shi, Xinglin Pan, Qiang Wang, Chengjian Liu, Xiaozhe Ren, Zhongzhe Hu, Yu Yang, Bo Li, and Xiaowen Chu. Schemoe: An extensible mixture-of-experts distributed training system with tasks scheduling. EuroSys '24, page 236–249, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704376. doi: 10.1145/3627703.3650083. URL https://doi.org/10.1145/3627703.3650083.

Mohammad Shoeybi et al. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019a.

Mohammad Shoeybi et al. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019b.

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

Siddharth Singh, Olatunji Ruwase, Ammar Ahmad Awan, Samyam Rajbhandari, Yuxiong He, and Abhinav Bhatele. A hybrid tensor-expert-data parallelism approach to optimize mixture-of-experts training. In *Proceedings of the 37th ACM International Conference on Supercomputing*, ICS '23, page 203–214, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700569. doi: 10.1145/3577193.3593704. URL https://doi.org/10.1145/3577193.3593704.

John Smith et al. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. Rotatekv: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations. *arXiv preprint arXiv:2501.16383*, 2025a. URL https://arxiv.org/abs/2501.16383.

Zunhai Su, Wang Shen, Linge Li, Zhe Chen, Hanyu Wei, Huangqi Yu, and Kehong Yuan. Akvq-vl: Attention-aware kv cache adaptive 2-bit quantization for vision-language models. *arXiv preprint arXiv:2501.15021*, 2025b.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023a.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023b. URL https://arxiv.org/abs/2306.11695.

Zhiqing Sun, Hongyin Li, Hao Zhang, Lei Zhang, Shuohang Wang, Jingjing Li, and Yang Liu. Spp: Sparsity-preserved parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2405.16057*, 2023c. URL https://arxiv.org/abs/2405.16057.

Aaquib Syed, Phillip Huang Guo, and Vijaykaarti Sundarapandiyan. Prune and tune: Improving efficient pruning techniques for massive language models. 2023.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *arXiv preprint arXiv:2407.13623*, 2024.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022. doi: 10.1145/3530811.

NLLB Team and Marta R. Costa-jussà. No language left behind: Scaling human-centered machine translation, 2022. URL https://arxiv.org/abs/2207.04672.

Lucas Theis and Matthias Bethge. Generative image modeling using spatial lstms. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/2b6d65b9a9445c4271ab9076ead5605a-Paper.pdf.

Maksim Timiryasov, Oleksii Hrinchuk, Oleksii Kuchaiev, and Boris Ginsburg. Baby llama: Efficiently distilling LLaMA to 4-bit. *arXiv preprint arXiv:2304.04148*, 2023. URL https://arxiv.org/abs/2304.04148.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, cs.CL, 2023b.

Dezhan Tu, Danylo Vashchilenko, Yuzhe Lu, et al. Vl-cache: Sparsity and modality-aware kv cache compression for vision-language model inference acceleration. *arXiv preprint arXiv:2410.23317*, 2024.

Lewis Tunstall, Leandro von Werra, Sheon Han, Anton Bakhtin, Thomas Scialom, Nisan Stiennon, Luke Zettlemoyer, and Teven Le Scao. Zephyr: Direct preference optimization as a reinforcement learning alternative for aligning language models. *arXiv preprint arXiv:2309.01375*, 2023. URL https://arxiv.org/abs/2309.01375.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, et al. An empirical study of mamba-based language models. *ArXiv*, cs.LG, 2024.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 2023.

Xin Wang, Zhongwei Wan, Arvin Hekmati, Mingyu Zong, Samiul Alam, Mi Zhang, and Bhaskar Krishnamachari. Iot in the era of generative ai: Vision and challenges. *arXiv preprint arXiv:2401.01923*, 2024a.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024b.

Yifan Wang et al. Entropy regularization for multi-modal model compression. *ICML*, 2022.

Yumeng Wang and Zhenyang Xiao. Loma: Lossless compressed memory attention. *arXiv preprint arXiv:2401.09486*, 2024.

Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, Xiaokun Wang, Yutuan Ma, Rui Hu, Shuicheng Yan, Han Fang, and Yahui Zhou. Skywork-moe: A deep dive into training techniques for mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2406.06563.

Yuxin Wen, Qingqing Cao, Qichen Fu, et al. Efficient vision-language models by summarizing visual tokens into compact registers. *arXiv preprint arXiv:2410.14072*, 2024.

Jiayun Weng, Yan Zhang, and Jing Liu. Transformer models and their application to reasoning tasks. *Journal of Machine Learning Research*, 24(1):1–28, 2023. Available at: https://jmlr.org/papers/volume24/.

Chiyu Wu, Yutai Geng, Xiang Li, Tianle Zhang, Zheng Zhang, Haoran Zhang, Xiang Zhang, Yizhe Zhang, Jianzhu Gao, Weizhu Chen, et al. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *arXiv preprint arXiv:2304.14402*, 2023a. URL https://arxiv.org/abs/2304.14402.

Xiaoxia Wu, Zhewei Yao, and Yuxiong He. Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats. *arXiv preprint arXiv:2307.09782*, 2023b.

Haocheng Xi, Yuxiang Chen, Kang Zhao, Kai Jun Teh, Jianfei Chen, and Jun Zhu. Jetfire: Efficient and accurate transformer pretraining with int8 data flow and per-block quantization. *arXiv preprint arXiv:2403.12422*, 2024.

Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. *arXiv preprint arXiv:2309.10285*, 2023a.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023b.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

Yi Xin, Siqi Luo, Xuyang Liu, Yuntao Du, Haodi Zhou, Xinyu Cheng, Christina Luoluo Lee, Junlong Du, Haozhe Wang, MingCai Chen, et al. V-petl bench: A unified visual parameter-efficient transfer learning benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Long Xing, Qidong Huang, Xiaoyi Dong, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

Jing Xiong, Gongye Liu, Lun Huang, Chengyue Wu, Taiqiang Wu, Yao Mu, Yuan Yao, Hui Shen, Zhongwei Wan, Jinfa Huang, et al. Autoregressive models in vision: A survey. *arXiv preprint arXiv:2411.05902*, 2024a.

Jing Xiong, Jianghan Shen, Fanghua Ye, Chaofan Tao, Zhongwei Wan, Jianqiao Lu, Xun Wu, Chuanyang Zheng, Zhijiang Guo, Lingpeng Kong, et al. Uncomp: Uncertainty-aware long-context compressor for efficient large language model inference. *arXiv preprint arXiv:2410.03090*, 2024b.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *arXiv preprint arXiv:2402.16880*, 2024.

Wei Xu, Jidong Li, Zhiqing Zhao, et al. Efficient fine-tuning and compression of large language models with low-rank adaptation and pruning. *arXiv preprint arXiv:2302.08582*, 2023.

Wei Xu et al. A survey on mixed-precision training for neural networks on cpus. *arXiv preprint arXiv:2210.09565*, 2022.

Yang You Xu et al. Colossal-ai: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*, 2021.

Huanrui Yang, Zhenyu Zhang, Yiran Chen, and Hai Li. Efficientdm: Robust and efficient quantization for diffusion models. *arXiv preprint arXiv:2303.09556*, 2023. URL https://arxiv.org/abs/2303.09556.

Jinghan Yao, Quentin Anthony, Aamir Shafi, Hari Subramoni, and Dhabaleswar K. DK Panda. Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 915–925, 2024. doi: 10.1109/IPDPS57955.2024.00086.

Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Quant-llm: Post-training quantization for large language models. *arXiv preprint arXiv:2308.07339*, 2023. URL https://arxiv.org/abs/2308.07339.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. Edgemoe: Fast on-device inference of moe-based large language models, 2023. URL https://arxiv.org/abs/2308.14352.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Mingshu Zhai, Jiaao He, Zixuan Ma, Zan Zong, Runqing Zhang, and Jidong Zhai. SmartMoE: Efficiently training Sparsely-Activated models through combining offline and online parallelization. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 961–975, Boston, MA, July 2023. USENIX Association. ISBN 978-1-939133-35-9. URL https://www.usenix.org/conference/atc23/presentation/zhai.

Honghe Zhang, XiaolongShi XiaolongShi, Jingwei Sun, and Guangzhong Sun. Structured pruning for large language models using coupled components elimination and minor fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1–12, 2024a.

Huaizheng Zhang, Ruiyuan Gao, Shangyang Sun, Vincent Y. F. Tan, and Bryan Kian Hsiang Low. Approximating softmax with relu for interpretable and efficient transformer architectures. *ArXiv*, cs.LG, 2023a.

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

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*, 2023c.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3013–3026, 2024b.

Nan Zhang, Yanchi Liu, Xujiang Zhao, Wei Cheng, Runxue Bao, Rui Zhang, Prasenjit Mitra, and Haifeng Chen. Pruning as a domain-specific llm extractor. *arXiv preprint arXiv:2405.06275*, 2024c.

Qi Zhang et al. Efficient eviction policies for multi-modal key-value stores. *CVPR*, 2022.

Wei Zhang et al. Guaranteed approximation bounds for mixed-precision neural operators. *arXiv preprint arXiv:2307.15034*, 2023d.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*, 2024d.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. 2023e. URL https://arxiv.org/abs/2306.14048.

Zhenyu Zhang, Huanrui Yang, Yiran Chen, and Hai Li. Dilatequant: Accurate and efficient diffusion quantization via weight dilation. *arXiv preprint arXiv:2306.03881*, 2023f. URL https://arxiv.org/abs/2306.03881.

Changsheng Zhao, Ting Hua, Yilin Shen, Qian Lou, and Hongxia Jin. Automatic mixed-precision quantization search of bert. *arXiv preprint arXiv:2112.14938*, 2021.

Jianfeng Zhao, Qi Liu, Rui Wang, and Kai Xu. Multi-stage knowledge distillation with collaborative refinement. *arXiv preprint arXiv:2401.12345*, 2024a. URL https://arxiv.org/abs/2401.12345.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *arXiv preprint arXiv:2310.19102*, 2023. URL https://arxiv.org/abs/2310.19102.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024b.

Xingyu Zheng, Xianglong Liu, Yichen Bian, Xudong Ma, Yulun Zhang, Jiakai Wang, Jinyang Guo, and Haotong Qin. Bidm: Pushing the limit of quantization for diffusion models. *arXiv preprint arXiv:2412.05926*, 2024a.

Xingyu Zheng, Xianglong Liu, Yichen Bian, Xudong Ma, Yulun Zhang, Jiakai Wang, Jinyang Guo, and Haotong Qin. Bidm: Pushing the limit of quantization for diffusion models. 2024b. URL https://arxiv.org/abs/2412.05926.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. URL https://arxiv.org/abs/1606.06160.