MeMoTune: A Measure and Moment-Driven Fine-Tuning Framework for Quantized Large Language Models

Yun Zhang[1,2], Xue Geng[2], Lizi Liao[3],
Jintong Sun[4], Minghe Yu[4], Ge Yu[1]

[1]School of Computer Science and Engineering, Northeastern University, China
[2]Institute for Infocomm Research, Agency for Science, Technology and Research, Singapore
[3]School of Computing and Information Systems, Singapore Management University, Singapore
[4]Software College, Northeastern University, China

Correspondence: geng_xue@i2r.a-star.edu.sg

Abstract

Quantizing large language models (LLMs) is essential for reducing memory and computational costs in natural language processing. Existing methods combine quantization with parameter-efficient fine-tuning but often fail to meet practical performance requirements. This paper introduces MeMoTune, a novel fine-tuning framework for quantized LLMs. By employing a measure and moment approach within a low-rank approximation framework in probability measure space, MeMoTune optimizes the objective function for superior fine-tuning results. The update process is further refined through scaled gradient, enhancing convergence efficiency and noise robustness. Experiments on tasks like text generation, summarization, and understanding show MeMoTune significantly outperforms state-of-the-art methods, *e.g.* fine-tuning Llama2-13B on GSM8K improves accuracy by 5.5%, while fine-tuning DeBERTaV3-base on CoLA of GLUE increases Matthews correlation by 1.7%. The code is publicly available at `github.com/hddyyyb/MeMoTune`.

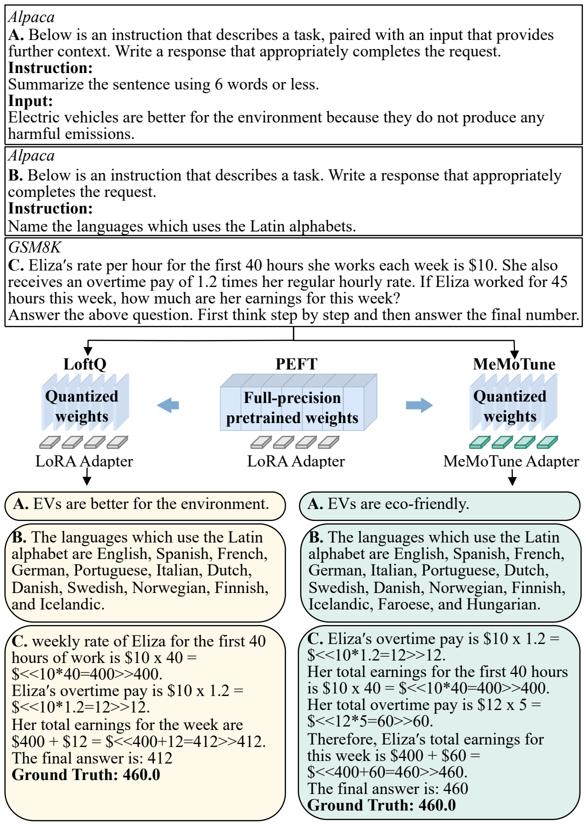

Figure 1: Comparison of MemoTune and LoftQ (Li et al., 2024) across the NLG task for Alpaca and GSM8K datasets. Our MeMoTune generates outputs that are: (A) more concise in summaries, (B) more comprehensive in enumerating answers to practical questions, and (C) more accurate in mathematical solutions.

1 Introduction

Large language models (LLMs) have transformed natural language processing (NLP), enabling significant advancements in various applications (Touvron et al., 2023a,b; AI@Meta, 2024; Lewis et al., 2019; Chung et al., 2024). Despite their success, the substantial memory and energy demands of LLMs limit their adoption, particularly on resource-constrained devices (Sze et al., 2017). Quantization has emerged as a crucial technique for reducing the computational and memory overhead of LLMs. By approximating high-precision weights with low-precision values, quantization effectively compresses models without retraining (Zhu et al., 2023; Frantar et al., 2022; Liu et al., 2023c; Xiao et al., 2023). For instance, 4-bit quantization methods (Banner et al., 2019; Sun et al., 2020) can reduce model size by up to 8× compared to full-precision (FP32) models. More recently, post-training quantization methods (Frantar et al., 2022; Yao et al., 2022; Liu et al., 2023c,a; Xiao et al., 2023) have demonstrated efficacy at 8-bit precision, as in SmoothQuant (Xiao et al., 2023), which enables lossless 8-bit weight and activation quantization. However, without retraining these methods often fail to maintain performance addresing tasks lower precisions than 8-bit (Wei et al., 2023; Lin et al., 2024).

Parameter-efficient fine-tuning (PEFT) has be-

Findings of the Association for Computational Linguistics: ACL 2025, pages 4036–4050
July 27 - August 1, 2025

come a key strategy for adapting full-precision LLMs to downstream tasks. By fine-tuning only a subset of parameters, PEFT achieves efficiency and performance gains (Hu et al., 2021; Lester et al., 2021; Li and Liang, 2021; Liu et al., 2023b). Huggingface's PEFT library offer tools for fine-tuning LLMs with minimal overhead (Mangrulkar et al., 2022). Low-Rank Adaptation (LoRA) (Hu et al., 2021), a leading PEFT method, introduces low-rank adapters to reduce computational costs. However, (Chen et al., 2022) demonstrates that PEFT methods exhibit limited effectiveness on medium-volume (1k–10k training samples) and large-volume (10k+ training samples) datasets. Additionally, experiments (Yang et al., 2024c) indicate that increasing fine-tuning data does not consistently improve results for a given task.

Recent studies integrate PEFT with quantization to enhance model performance while reducing memory overhead. QLoRA (Dettmers et al., 2024) and IR-QLoRA (Qin et al., 2024) fine-tune LoRA parameters while keeping the pre-trained base model's 4-bit parameters frozen. LoftQ (Li et al., 2024) leverages singular value decomposition (SVD) to initialize LoRA, improving performance and addressing quantization challenges. These methods inherit the limitations of full-precision PEFT, while quantization-induced degradation further impacts performance.

In this paper, we propose a novel framework, **Measure and Moment-driven fine-Tuning** (**MeMoTune**), for quantized LLMs. MeMoTune models trainable weights as a high-dimensional Gaussian distribution within the LoRA architecture and optimizes the objective function, with a theoretical analysis of this process, thereby improving performance and addressing its limitations on large datasets. Additionally, MeMoTune derives distribution parameter gradients while introducing a scale into the update process, enabling efficient distribution parameter updates, improving convergence, and enhancing noise resilience. We evaluate MeMoTune across a range of NLP tasks, including natural language generation (NLG), summarization, and natural language understanding (NLU). Experimental results demonstrate that MeMoTune outperforms state-of-the-art (SOTA) methods significantly. For example, fine-tuning Llama2-13B on GSM8K dataset improves accuracy by 5.5%, while fine-tuning DeBERTaV3-base on CoLA of GLUE improves Matthews correlation by 1.7%. These results highlight MeMoTune's potential for deploying quantized LLMs in resource-constrained environments. Figure 1 showcases MeMoTune's improvements over LoftQ (Li et al., 2024), demonstrating its strong reasoning capabilities. In summary, our contributions are threefold:

- We propose MeMoTune, leveraging the measure and moment approach to model trainable weights as a Gaussian distribution in LoRA, optimizing fine-tuning for quantized LLMs.
- We derive distribution parameter gradients while introducing a scale to enhance convergence efficiency and noise robustness.
- Experiments show that our method achieves SOTA results across NLP tasks with significant accuracy and efficiency gains.

2 Related work

2.1 Quantization

Quantization reduces the precision of model parameters and activations by storing them in low-bit formats, which reduces memory and computational needs. There are two main approaches to quantization: Quantization-Aware Training (QAT) (Shin et al., 2023; Zhu et al., 2023) and Post-Training Quantization (PTQ) (Liu et al., 2023a,c; Xiao et al., 2023). QAT incorporates quantization during training, allowing the model to adapt to quantization errors, often resulting in higher accuracy than PTQ. For instance, QAT-based method QFD (Zhu et al., 2023) trains a quantized representation as the teacher, and quantize the network using knowledge distillation. However, QAT requires retraining, which can be computationally expensive.

In contrast, PTQ quantizes pre-trained base models without additional training, making it more efficient. Methods like Gptq (Frantar et al., 2022), a one-shot weight quantization approach using approximate second-order information, and Zeroquant (Yao et al., 2022), which applies group-wise weight and token-wise activation quantization, effectively reduce quantization error and maintain hardware acceleration. However, PTQ may still result in significant performance degradation.

2.2 Parameter-Efficient Fine-Tuning (PEFT)

PEFT reduces computational costs by updating only a small subset of model parameters during fine-tuning. LoRA (Hu et al., 2021), a foundational PEFT technique, introduces low-rank matrices, which is called **Adapter**, to represent the necessary changes from pre-trained to fine-tuned weights, avoiding full matrix updates. It enables efficient adaptation with minimal parameter updates. However, LoRA's deterministic linear approximation limits robustness, making it sensitive to noisy data. Other methods, such as Prompt Tuning (Lester et al., 2021), Prefix-Tuning (Li and Liang, 2021), and P-Tuning (Liu et al., 2023b), fine-tune input prompts or attention-specific tokens, keeping core model parameters unchanged.

Recent work improves LoRA's adaptability and efficiency. DoRA (Mao et al., 2024) prunes less impactful components, MELoRA (Ren et al., 2024) trains mini-LoRAs across dimensions, and PRoLoRA (Wang et al., 2024) shares parameter blocks within layers. Tunable mask filtering parameters have also been proposed to enhance fine-tuning performance (Nikdan et al., 2024; Xu and Zhang, 2024; Lu et al., 2024; Zhang et al., 2024). Laplace-LoRA (Yang et al., 2024a) estimates uncertainty via Laplace approximation without altering LoRA-trained weights, focusing on calibration rather than accuracy. However, these methods are not designed for quantized LLMs, limiting their effectiveness. Moreover, (Chen et al., 2022) shows their limitations in medium-volume and large-volume datasets, while (Yang et al., 2024c) finds that more fine-tuning data does not always enhance performance.

2.3 PEFT for Quantized LLMs

PEFT has been applied to improve the performance of quantized LLMs. QLoRA (Dettmers et al., 2024) freezes the quantized pre-trained model and fine-tunes the LoRA adapter. It introduces k-bit NormalFloat (NFk), Double Quantization, and Paged Optimizers to save memory without sacrificing performance. LoftQ (Li et al., 2024) improves the combination of quantization and fine-tuning in LoRA by minimizing the gap between original and N-bit quantized weights plus low-rank approximations using alternating quantization and SVD. To address information loss in quantization, IR-QLoRA (Qin et al., 2024) proposes statistics-based calibration and entropy maximization to retain original information, along with fine-tuning to enhance LoRA's information recovery. These hybrid methods preserve pre-quantization data characteristics by training selected full-precision parameters. However, they still inherit the limitations of quantization and PEFT, additionally struggling with handling complex data and remaining susceptible to noise during fine-tuning.

3 Background

3.1 Low-Rank Adaptation (LoRA)

Given a weight matrix $\boldsymbol{W} \in \mathbb{R}^{d\times t}$ in a neural network, LoRA approximates the fine-tuning updates by decomposing the update matrix $\Delta\boldsymbol{W}$ into two low-rank matrices (adapters): $\Delta\boldsymbol{W} = \boldsymbol{AB}$, where $\boldsymbol{A} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r\times t}$ are adapters, r is a hyperparameter representing the rank of the decomposition, typically chosen such that $r \ll \min(d, t)$. During the fine-tuning process, the weight matrix $\boldsymbol{W}$ is updated as:

$$\boldsymbol{W}_{new} = \boldsymbol{W} + \Delta\boldsymbol{W} = \boldsymbol{W} + \boldsymbol{AB}. \quad (1)$$

This formulation keeps the pre-trained weights $\boldsymbol{W}$ fixed when training the model on a downstream task with the adapted layers, updating only $\boldsymbol{A}$ and $\boldsymbol{B}$ to capture task-specific information. This improves parameter efficiency by reducing the number of trainable parameters from dt to $dr + rt$, significantly saving memory and computation costs.

3.2 Quantization

Our research considers two quantization methods: uniform quantization and quantile quantization. Using the quantization function $q(\cdot)$, the full-precision weights $\boldsymbol{W}$ are quantized into a k-bit representation $\boldsymbol{W}_q$. In this paper, $k \in \{2, 4\}$. If $q(\cdot)$ represents uniform quantization over the range $[\boldsymbol{a}, \boldsymbol{b}]$, then $\boldsymbol{W}_q = q(\boldsymbol{W}) = \boldsymbol{a} + \frac{\boldsymbol{b}-\boldsymbol{a}}{2^k-1}\lfloor(2^k-1)\frac{\boldsymbol{W}-\boldsymbol{a}}{\boldsymbol{b}-\boldsymbol{a}}\rceil$, where $\lfloor\cdot\rceil$ denotes rounding to the nearest integer. However, uniform quantization suffers from the issue of data difference disappearance in densely populated regions due to its fixed interval mapping. To mitigate this problem, the non-linear method Quantile Quantization (Dettmers et al., 2022) divides data by finding some quantiles and maps them in the same interval to the same value, which allocates more representation capacity to regions with dense data distributions. In our approach, we adopt k-bit *NormalFloat Quantization* (NFk) (Dettmers et al., 2024) as an implementation of Quantile Quantization. Further details are provided in Appendix A.

4 Methodology

We propose a measure and moment-driven fine-tuning framework for quantized LLMs (MeMoTune). This approach explicitly models and learns the pairwise correlations between weights during training, leading to improved performance significantly. Figure 2 is a brief illustration of MeMoTune.

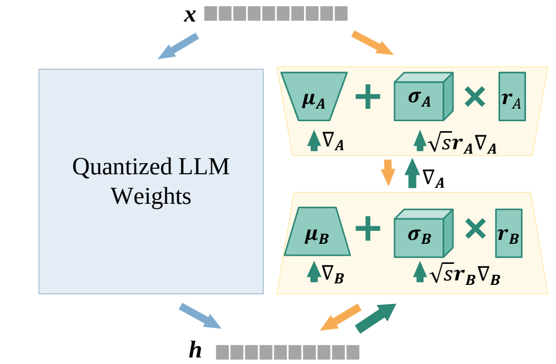

Figure 2: The overview of MeMoTune. The model optimizes parameter distributions and designs a back-propagation method, where ∇ represents the gradients.

Module	Layer 17		Layer 20	
	Full-Prec.	NF4	Full-Prec.	NF4
self_attn.q_proj	1.7120	1.8680	1.6200	1.7774
self_attn.k_proj	3.6261	3.4212	3.5204	3.4119
self_attn.v_proj	0.1604	0.1607	0.1674	0.1613
self_attn.o_proj	0.0146	0.0163	0.0163	0.0194
mlp.gate_proj	0.1686	0.1880	0.1699	0.1861
mlp.act_fn	0.0352	0.0403	0.0340	0.0400
mlp.up_proj	0.1194	0.1301	0.1262	0.1351
mlp.down_proj	0.0273	0.0349	0.0327	0.0358

Table 1: Layer-wise activation variance of Llama2-7B on GSM8K under full-precision and NF4 quantization.

We train the distribution of model parameters from the perspective of probability measure and design the corresponding back-propagation method.

For a domain set $\mathcal{X}$ and a label set $\mathcal{Y}$, let $f : \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y}$ represent a neural network. Denote by $y_{\boldsymbol{x}}$ the true label of instance $\boldsymbol{x}$ and by L the smooth loss function. Our training objective can be expressed as the following optimization problem.

$$\min_{\boldsymbol{A},\boldsymbol{B}} \mathbb{E}_{\boldsymbol{x}}[L(f(\boldsymbol{x}, \widehat{\boldsymbol{W}} + \boldsymbol{A}\boldsymbol{B}), y_{\boldsymbol{x}})], \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r\times t}$ are parameter matrices to be fine-tuned, $\widehat{\boldsymbol{W}} \in \{q(\boldsymbol{W})\} \subseteq \mathbb{R}^{d\times t}$, and $\{q(\boldsymbol{W})\}$ represents the possible value set of the quantized parameter. In the following part of this paper, we will abbreviate $L(f(\boldsymbol{x}, \widehat{\boldsymbol{W}} + \boldsymbol{A}\boldsymbol{B}), y_{\boldsymbol{x}})$ as $L(f, y)$.

4.1 Measure and Moment-Based Objective

SOTA fine-tuning methods struggle on medium- and large-volume datasets (Chen et al., 2022), and increasing fine-tuning data does not always yield better results (Yang et al., 2024c), potentially due to noise introduced during training (Wei et al., 2022).

To better understand the role of noise in quantized models, we analyzed the layer-wise activation variance of Llama2-7B on the GSM8K dataset under two settings: (1) full-precision and (2) NF4 quantization. Taking layers 17 and 20 as representative examples, the results presented in Table 1 reveal notable variance shifts under quantization, which could be interpreted as noise-like perturbations that may contribute to performance degradation. In addition, models suffer from catastrophic forgetting (Kirkpatrick et al., 2017), as distribution shifts between pre-training and fine-tuning data impair reasoning abilities.

To address this, we propose training parameter distributions instead of single estimates, enabling the model to capture weight correlations and introduce noise, which enhances regularization and improves robustness and generalization (Blundell et al., 2015; Gal and Ghahramani, 2016), while simultaneously preserving the benefits of large models. Thus, we address the problem through probability measures on parameter distributions.

According to the measure and moment approach (Jasour, 2019), the unconstrained optimization with the random variable z:

$$P^* = \min_{z\in\Omega} p(z), \tag{3}$$

where $\Omega = \mathbb{R}^n$, can be transformed into an optimization problem in terms of probability distributions (measures) with the random variable ν, i.e.,

$$P^*_\nu = \min_{\nu\in\mathcal{M}(\Omega)} \mathbb{E}_\nu[p(z)] = \min_{\nu\in\mathcal{M}(\Omega)} \int_\Omega p(z)d\nu, \tag{4}$$

where ν is the probability measure associated with z subject to $\int_\Omega d\nu = 1$, and $\mathcal{M}(\Omega)$ denotes the space of measures supported on Ω.

Applying this measure-theoretic perspective, we treat the adapter parameters $\boldsymbol{A}$ and $\boldsymbol{B}$ as random variables, akin to z in Eq. (3). Thus, the optimization problem in Eq. (2) can be transformed into:

$$\min_{\boldsymbol{\beta_A},\boldsymbol{\beta_B}} \mathbb{E}_{\boldsymbol{\beta_A},\boldsymbol{\beta_B}}[L(f, y)], \tag{5}$$

where $\boldsymbol{\beta_A}$ and $\boldsymbol{\beta_B}$ are the probability measures of trainable parameters $\boldsymbol{A}$ and $\boldsymbol{B}$. This formulation requires finding the optimal probability measures, specifically the probability density functions of $\boldsymbol{A}$ and $\boldsymbol{B}$, that minimize the expected loss. For ease of calculation, we assume that $\boldsymbol{A}$ and $\boldsymbol{B}$ follow Gaussian distribution, *i.e.* $\boldsymbol{A} \sim \mathcal{N}(\boldsymbol{\mu_A}, \boldsymbol{\Sigma_A})$, $\boldsymbol{B} \sim \mathcal{N}(\boldsymbol{\mu_B}, \boldsymbol{\Sigma_B})$, where $\boldsymbol{\Sigma_A}$ and $\boldsymbol{\Sigma_B}$ capture the correlation among different elements within $\boldsymbol{A}$ and $\boldsymbol{B}$. Since the parameters are not updated directly, it is crucial to clarify how their distributions are refined, which involve updating the expectation and covariance.

4.2 Distribution Updating

Our goal is to derive the gradients of distribution parameters, *i.e.*, expectation and covariance. During training, we can easily get the gradients $\nabla_{\boldsymbol{A}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)]$ and $\nabla_{\boldsymbol{B}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)]$. Establishing their relationship with distribution parameter gradients allows us to derive a method for updating the distribution.

We begin by analyzing the variable $\boldsymbol{A}$. To facilitate the derivation, we introduce an auxiliary variable $\boldsymbol{r}$:

$$\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \frac{1}{\sqrt{s}}\boldsymbol{I}), \tag{6}$$

where $s \geq 100$ is a hyperparameter representing the scale size. The scaling factor s helps reduce excessive noise from the standard normal distribution, preventing adverse effects on large models. We model $\boldsymbol{A}$ with the expectation and covariance matrix by introducing the random noise $\boldsymbol{r}$ as follows.

$$\boldsymbol{A} = \boldsymbol{\mu}_{\boldsymbol{A}} + \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}}\boldsymbol{r}, \tag{7}$$

where $\boldsymbol{\mu}_{\mathbf{A}}$ denotes the learnable expectation parameter and $\boldsymbol{\Sigma}_{\boldsymbol{A}_r}$ denotes the learnable covariance matrix parameter. Under this formulation, we derive $\boldsymbol{\Sigma}_{\boldsymbol{A}} = \frac{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}}{s}$, but there is no need to explicitly compute or store it.

To obtain $\boldsymbol{A}$ after each sampling of $\boldsymbol{r}$, we infer the gradients of $\boldsymbol{\mu}_{\boldsymbol{A}}$ and $\sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}}$ from the gradient of $\boldsymbol{A}$. The specific relationships are stated in the following theorem, with its proof provided in Appendix B.

Theorem 1. *For the random variable* $\boldsymbol{A} = \boldsymbol{\mu}_{\boldsymbol{A}} + \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}}\boldsymbol{r}$, $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \frac{1}{\sqrt{s}}\boldsymbol{I})$ *with* $\boldsymbol{\mu}_{\boldsymbol{A}} \in \mathbb{R}^{d\times r}$, $\sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}} \in \mathbb{R}^{d\times r\times K}$ *and* $s \in \mathbb{R}^1$, *we have:*

$$\nabla_{\boldsymbol{\mu}_{\boldsymbol{A}}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)] = \nabla_{\boldsymbol{A}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)]. \tag{8}$$

$$\nabla_{\sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)] = \boldsymbol{r}\nabla_{\boldsymbol{A}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)]. \tag{9}$$

A large scale s results in a smaller variance, narrowing the range of $\boldsymbol{r}$. To prevent the gradient from becoming too small (vanishing), we scale up the gradient as follows:

$$\nabla_{\sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)] := \sqrt{s}\boldsymbol{r}\nabla_{\boldsymbol{A}}\mathbb{E}_{\boldsymbol{x}}[L(f,y)]. \tag{10}$$

Thus, $(\sqrt{s}\boldsymbol{r}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Consequently, the scaled gradient effectively updates distribution parameters, improving convergence efficiency.

The same derivation applies to $\boldsymbol{B}$. Through this process, we obtain the gradients for the distribution parameters, which are then used to train the distributions of $\boldsymbol{A}$ and $\boldsymbol{B}$, enabling us to solve the objective in Eq. (5).

4.3 Lower Memory and Time Cost

To conserve computing resources, we approximate the square root of the covariance matrix parameter with a low-rank matrix representation: $\sqrt{\boldsymbol{\Sigma}_{\boldsymbol{A}_r}} \approx \boldsymbol{\sigma}_{\boldsymbol{A}}$, and $\sqrt{\boldsymbol{\Sigma}_{\boldsymbol{B}_r}} \approx \boldsymbol{\sigma}_{\boldsymbol{B}}$, where $\boldsymbol{\sigma}_{\boldsymbol{A}} \in \mathbb{R}^{d\times r\times K}$, $\boldsymbol{\sigma}_{\boldsymbol{B}} \in \mathbb{R}^{r\times t\times K}$, $K \ll \min(dr)$. On this basis, the forward pass changes from $(\boldsymbol{W}_q + \boldsymbol{AB})\boldsymbol{x}$ to $(\boldsymbol{W}_q + (\boldsymbol{\mu}_{\boldsymbol{A}} + \boldsymbol{\sigma}_{\boldsymbol{A}}\boldsymbol{r})(\boldsymbol{\mu}_{\boldsymbol{B}} + \boldsymbol{\sigma}_{\boldsymbol{B}}\boldsymbol{r}))\boldsymbol{x}$. The corresponding backward pass updates the parameters according to Eq. (8) and Eq. (9).

Therefore, for each layer, the memory cost of the base model parameters is $\mathcal{O}(\frac{dtk}{32})$. For the trainable part, we need to store the expectation and covariance parameters. Specifically, the memory cost of expectation parameters $\boldsymbol{\mu}_{\boldsymbol{A}}$ and $\boldsymbol{\mu}_{\boldsymbol{B}}$ is $\mathcal{O}((k+d)r)$, while the memory cost of the approximate covariance parameters $\boldsymbol{\sigma}_{\boldsymbol{A}}$ and $\boldsymbol{\sigma}_{\boldsymbol{B}}$ is $\mathcal{O}((k+d)rK)$. To ensure a low memory cost, we constrain the value of K to be relatively small, i.e., $K \leq 4$. This restriction results in low ranks for $\boldsymbol{\Sigma}_{\boldsymbol{A}_r}$ and $\boldsymbol{\Sigma}_{\boldsymbol{B}_r}$, implying that the majority of trainable parameters are independent of each other. This approach is both reasonable and does not compromise the performance of the method. The influence of K on performance is further evaluated in Section 5.6.

Considering the time complexity, during training, both the expectation and covariance of parameters $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated simultaneously, resulting in a time complexity of $\mathcal{O}((K+1)(rt+rd))$. In inference, while dequantization occurs during the forward pass, it typically adds only a constant cost $\mathcal{O}(c)$ to the standard matrix multiplication. Therefore, the overall time complexity is $\mathcal{O}(td + (K+1)(rt+rd) + c)$. In comparison to methods such as QLoRA or LoftQ, the additional complexity arises from the parameter K. By constraining K to a small value, our method achieves a lower time complexity.

5 Experiments

5.1 Experimental Setup

Datasets and Metrics We evaluate MeMoTune on three fundamental NLP tasks: NLG, text summarization, and NLU. These tasks assess the model's ability to generate, condense, and understand text effectively. We summarize the datasets, architectures, and metrics for each task.

For NLG, we follow the QLoRA and IR-QLoRA fine-tuning framework, training Llama (Touvron et al., 2023a) and Llama2 (Touvron et al., 2023b) on Alpaca (Taori et al., 2023), Hh-rlhf (Bai et al., 2022), and GSM8K (Cobbe et al., 2021), and additionally fine-tune Mistral (Jiang et al., 2023) and Qwen2.5 (Yang et al., 2024b; Team, 2024) on GSM8K. Perplexity (Bengio et al., 2003) is used as the evaluation metric for Alpaca and Hh-rlhf, while accuracy is used for GSM8K.

For the summarization task, following the fine-tuning architecture of LoftQ, T5 and BART are fine-tuned on DialogSum (Chen et al., 2021) and BillSum (Kornilova and Eidelman, 2019) datasets, evaluated using ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) metrics.

For NLU tasks, following the fine-tuning architecture of LoftQ and IR-QLoRA, we fine-tune DeBERTaV3-base on the GLUE benchmark (Wang et al., 2018), with evaluation metrics including Matt, Acc, f1 and P/S Corr. We also evaluate Llama and Llama2 on the MMLU (Hendrycks et al., 2021) dataset, using accuracy as the evaluation metric.

Following (Chen et al., 2022), NLG and summarization fine-tuning occur on large-volume datasets (10k+ samples), while NLU fine-tuning is on medium-volume datasets (1K–10K samples). Base model details are provided in Appendix C.1, and additional information on datasets and evaluation metrics can be found in Appendix C.2.

Training Details For NLG and text summarization tasks, the base models are quantized to 4 bits using NFk (Dettmers et al., 2024; Qin et al., 2024; Li et al., 2024). Adapter ranks are set to 16 and 32 for Alpaca and Hh-rlhf, 64 for GSM8K in NLG tasks, and 16 for summarization tasks. For NLU tasks on GLUE, both NFk and uniform quantization compress the base models to 2 and 4 bits, with adapter ranks of 16 and 32. The fine-tuning settings for Llama and Llama2 on MMLU align with those used for fine-tuning Llama and Llama2 on Alpaca for the NLG task. Model evaluation is performed every 500 training steps, with median results reported over five runs using different random seeds. The adapter rank for all LoRA-based baselines remains consistent with MeMoTune. Detailed configurations are in Appendix C.3, and baselines adhere to the settings specified in their respective papers. Best results are in **bold**.

We initialize expectations $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ using an SVD-based method inspired by LoftQ (Li et al., 2024). The adapters are defined as $\boldsymbol{\mu}_A = [\sqrt{\boldsymbol{\lambda}_i}\boldsymbol{u}_i, \dots, \sqrt{\boldsymbol{\lambda}_r}\boldsymbol{u}_r]$ and $\boldsymbol{\mu}_B = [\sqrt{\boldsymbol{\lambda}_i}\boldsymbol{v}_i, \dots, \sqrt{\boldsymbol{\lambda}_r}\boldsymbol{v}_r]$, where $\boldsymbol{\lambda}_i$ are the singular values of the residual weight matrix $\boldsymbol{W} - q(\boldsymbol{W})$, and $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ are the left and right singular vectors. Covariances are initialized to $\mathbf{0}$.

5.2 Baselines

We compare our method with SOTA approaches and a **No Tuning** baseline (a full-precision model without fine-tuning). The details are as follows:

PEFT Methods They are full-precision methods that train only a subset of parameters. **LoRA** (Hu et al., 2021) reduces trainable parameters by introducing low-rank adaptation matrices while keeping the base model frozen, enabling efficient fine-tuning. **Prefix Tuning (PT)** (Li and Liang, 2021) prepends trainable continuous vectors to the key and value projections in the attention layers.

Quantization Plus PEFT These methods fine-tune a quantized model with PEFT for efficiency. **QLoRA** (Dettmers et al., 2024) combines k-bit quantization with LoRA, reducing memory while preserving performance. **IR-QLoRA** (Qin et al., 2024) improves quantized LLM accuracy via integrating LoRA with information retention techniques. **LoftQ** (Li et al., 2024) enhances QLoRA by optimizing adapter initialization.

5.3 NLG Task

We evaluate MeMoTune on the Alpaca and Hh-rlhf datasets, with results for adapter rank 16 presented in Table 2. MeMoTune consistently outperforms SOTA methods, except for fine-tuning Llama-7B, where full-precision LoRA yields better results. This suggests that optimizing parameter distributions in the measurement space and scaling gradients enhance performance across various inputs. PT fails to converge, while LoRA performs better, reinforcing the effectiveness of fine-tuning within the LoRA framework. On the ALpaca dataset, No Tuning shows better results than on the Hh-rlhf dataset, as pre-trained Llama and Llama2 can adapt well to this dataset. Moreover, on the more complex Hh-rlhf dataset, MeMoTune achieves significant improvements over baselines, underscoring its effectiveness in complex real-world scenarios. Additional results for adapter rank 32 are provided

Method	Alpaca				Hh-rlhf	
	Llama		Llama2		Llama	Llama2
	7B	13B	7B	13B	7B	7B
No Tuning	4.40	4.79	4.24	4.69	8.29	8.80
LoRA	2.80	**2.52**	2.85	2.63	2.16	2.30
PT	3.02	2.76	2.90	2.64	4.39	4.28
QLoRA	2.86	2.70	2.88	2.64	2.17	2.32
IR-QLoRA	2.93	2.82	2.87	2.78	1.98	2.02
LoftQ	2.79	2.58	2.86	2.53	2.22	2.24
MeMoTune	**2.76**	2.54	**2.79**	**2.51**	**1.83**	**1.89**

Table 2: Results of Llama and Llama2 on Alpaca and Hh-rlhf.

Method	Llama		Llama2		Mistral	Qwen2.5
	7B	13B	7B	13B	7B	7B
No Tuning	11.0	17.8	14.6	28.7	52.2	69.4
LoRA	32.8	36.6	36.9	43.1	57.0	**83.0**
QLoRA	30.3	37.4	35.1	39.9	56.7	79.6
LoftQ	34.8	43.7	35.0	45.0	57.9	80.8
MeMoTune	**35.8**	**44.2**	**40.9**	**50.5**	**59.0**	81.3

Table 3: Results of Llama, Llama2, Mistral and Qwen2.5 on GSM8K.

in Appendix D.1, where MeMoTune continues to deliver the best performance.

To evaluate problem-solving capabilities, we fine-tune Llama, Llama2, Mistral and Qwen2.5 on the GSM8K dataset, with results in Table 3. Baselines including PT and IR-QLoRA are excluded due to the lack of GSM8K implementations. Baseline results are sourced from LoftQ (Li et al., 2024) (Llama2 fine-tuned with LoftQ and QLoRA), and our training (other cases). MeMoTune consistently achieves the best performance among quantized fine-tuning methods, outperforming QLoRA and LoftQ across all models. It performs particularly excelling on Llama2, with accuracy gains of 4.1% (7B) and 5.5% (13B) over LoftQ. These results demonstrate MeMoTune's robustness and superior performance in fine-tuning quantized models on large-volume datasets.

5.4 Summarization Task

We fine-tune T5 and BART on the DialogSum and BillSum datasets to evaluate our method for text summarization. During baseline training, we observe that QLoRA outperforms LoRA for fine-tuning T5 on DialogSum, as shown in Figure 3. It is consist with QLoRA's findings that lower precision of more parameters results in the better performance. We hypothesize that this phenomenon arises because quantization acts as a form of regularization that helps prevent overfitting, a benefit especially useful in tasks like summarization.

Table 4 presents the text summarization results, showing that MeMoTune consistently outperforms

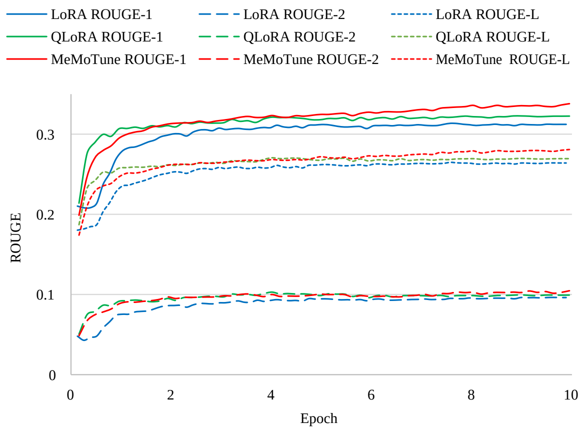

Figure 3: Fine-tuning T5 using LoRA, QLoRA, and MeMoTune on DialogSum.

baselines on DialogSum and BillSum when fine-tuning T5, BART-base, and BART-large. Among the baselines, LoftQ and IR-QLoRA excel in different scenarios. IR-QLoRA outperforms LoftQ on both datasets for fine-tuning T5 and BART-base, ranking just behind MeMoTune, while LoftQ performs better on BART-large. Full-precision methods do not surpass quantized ones, highlighting the benefits of quantization in summarization. These results confirm its ability to enhance quantized fine-tuning by optimizing parameter distributions in the measurement space, ensuring consistency across datasets and model sizes.

5.5 NLU Task

We evaluate MeMoTune's ability to fine-tune extremely low-bit models. Table 5 presents the results of fine-tuning 2-bit DeBERTaV3-base on the GLUE corpus (CoLA, MRPC, and STS-B), with baseline values from the LoftQ paper. IR-QLORA has not yet implemented the GLUE task for DeBERTaV3-base. MeMoTune consistently outperforms LoftQ in all cases, showing stable and superior performance. Notably, on CoLA with NF2 quantization and rank 32, MeMoTune achieves a Matthews correlation 1.7% higher than LoftQ, highlighting improved fine-tuning. Additional results for 4-bit DeBERTaV3-base in Appendix D.2 further confirm MeMoTune's effectiveness. By optimizing parameter distributions, MeMoTune excels in fine-tuning 2-bit and 4-bit models on medium-volume datasets.

The MMLU accuracy of fine-tuning Llama and Llama2 on the Alpaca dataset, with adapter rank 16, is shown in Table 6, with baseline values sourced from QLoRA (Dettmers et al., 2024) (QLoRA) and

Method	DialogSum			BillSum		
	T5	BART-base	BART-large	T5	BART-base	BART-large
No Tuning	20.4/5.0/17.3	17.4/2.7/15.6	17.5/2.7/15.8	15.1/7.0/12.5	3.6/0.5/3.2	6.6/1.9/5.8
PT	29.5/8.5/25.2	36.9/12.3/30.4	39.9/15.7/33.6	21.8/16.6/20.8	22.3/17.2/21.6	23.4/18.8/22.7
LoRA	31.2/9.7/26.4	39.0/14.0/32.2	41.4/17.7/35.6	23.5/18.6/22.5	23.6/18.9/22.9	24.5/20.5/24.0
QLoRA	32.3/9.9/27.0	38.1/14.3/32.5	41.8/17.8/35.6	23.7/18.8/22.8	23.8/19.3/23.2	24.3/20.2/23.7
LoftQ	33.6/9.7/28.7	38.9/14.6/32.4	42.2/18.3/36.0	23.7/18.8/22.9	23.8/19.3/23.1	24.4/20.4/23.9
IR-QLoRA	36.6/12.5/30.2	39.3/14.9/32.6	41.5/18.0/35.5	23.4/18.5/22.4	23.6/18.8/22.9	24.0/19.7/23.4
MeMoTune	**36.7/12.5/30.4**	**39.5/15.0/32.8**	**42.5/18.8/36.4**	**23.8/18.9/23.0**	**23.9/19.5/23.3**	**24.5/20.7/24.3**

Table 4: Results of T5, BART-base and BART-large on DialogSum and BillSum, evaluated with ROUGE-1/2/L (%).

Quant	Rank	Method	CoLA Matt	MRPC Acc/f1	STS-B P/S Corr
NF	16	LoftQ	37.4	83.8/88.6	87.1/86.9
		MeMoTune	**38.6**	**84.6/89.0**	**87.7/87.4**
	32	LoftQ	47.5	83.6/87.2	87.5/87.0
		MeMoTune	**49.2**	**84.6/89.2**	**88.2/87.9**
Unif	16	LoftQ	59.1	87.0/90.6	87.9/87.6
		MeMoTune	**59.6**	**87.8/91.4**	**88.2/87.8**
	32	LoftQ	60.5	87.5/91.2	89.5/89.2
		MeMoTune	**61.3**	**88.0/91.4**	**89.9/89.7**

Table 5: Results of 2-bit DeBERTaV3-base on GLUE.

Method	Llama		Llama2	
	7B	13B	7B	13B
No Tuning	35.1	43.3	44.3	55.3
PT	36.8	45.3	45.4	54.6
LoRA	40.6	47.4	46.4	**57.3**
QLoRA	38.8	47.8	44.4	51.6
LoftQ	39.1	48.5	44.7	53.5
IR-QLoRA	40.6	49.3	46.1	54.4
MeMoTune	**40.9**	**50.6**	**46.4**	56.6

Table 6: MMLU results of Llama and Llama2 on Alpaca.

IR-QLoRA (Qin et al., 2024) (No Tuning and IR-QLoRA) reports, compared to our training (LoRA, LoftQ, MeMoTune). MeMoTune achieves the highest accuracy among methods using quantized base models, outperforming IR-QLoRA by 1.3% on Llama-13B and 2.2% on Llama2-13B. This highlights its superior performance, particularly for larger models. Additional results for adapter rank 32 are provided in Appendix D.3. These results indicate that MeMoTune is highly effective on large-volume datasets.

5.6 Ablation Study

Effect of K. Building on prior results, training the parameter distributions has demonstrated effectiveness. Next, we explore the covariance within this distribution by comparing the effects of different K on the experimental outcomes. We fine-tune T5 and BART on DialogSum, with low-rank adapters of rank 16, and record the ROUGE-L scores for various K, as shown in Figure 4.

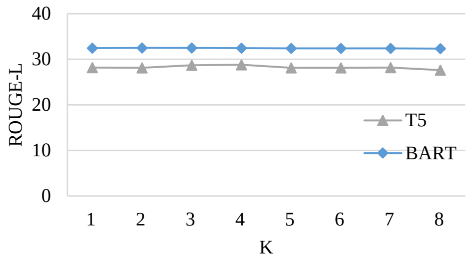

Figure 4: Fine-tuning T5 and BART on DialogSum.

We found that K has little effect on model performance, with results fluctuating within an acceptable range. This indicates that the elements in the parameter matrix are nearly independent, allowing a low-rank covariance matrix to achieve good results. However, as K increases (*e.g.*, $K = 8$), memory and computation costs rise significantly, making it difficult to achieve optimal performance. Therefore, we select $K \in \{1, 2, 4\}$ for this study. See Appendix C.3 for specific settings.

5.7 Loss Convergence in Fine-Tuning

We fine-tune the 4-bit Llama2-7B initialized with LoftQ by QLoRA and MeMoTune on the Hh-rlhf (Bai et al., 2022) dataset, where the parameters of the former follow the settings of LoftQ (Li et al., 2024) and both methods shuffle the data before each epoch of training. The changes in model performance are shown in Figure 5.

With LoftQ, training loss drops sharply at the start of each epoch but gradually increases over time. In contrast, MeMoTune mitigates this issue, achieving a faster perplexity reduction. We suspect this is due to LoftQ training parameters within the LoRA framework, which, owing to its low-rank nature, may struggle with complex data. By modeling parameter distributions, MeMoTune enhances robustness, generalization, and mitigates catastrophic forgetting in large models. Notably, further reducing the learning rate leads to earlier

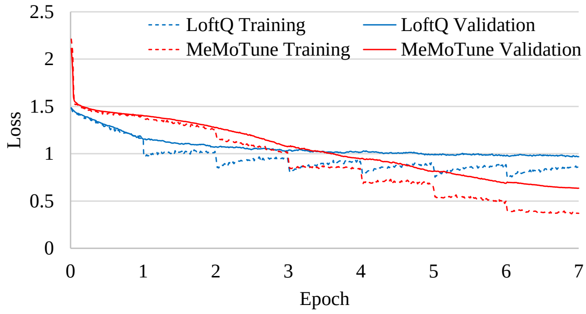

Figure 5: Fine-tuning Llama2-7B using LoftQ and MeMoTune on Hh-rlhf.

Noise Level	LoftQ	MeMoTune
0.00 (No Noise)	36.98	41.08
0.01	34.95	37.83
0.05	22.74	25.63
0.10	11.07	14.86
0.15	6.60	6.97
0.20	3.41	4.09

Table 7: Accuracy under varying noise levels on GSM8K.

convergence with worse performance, so we select an initial learning rate of 0.01 with a cosine decay schedule, as detailed in Section 5.1.

5.8 Evaluating Noise Robustness

To further demonstrate the noise robustness of our proposed method, we conducted additional experiments of fine-tuning Llama2-7B on the GSM8K dataset by injecting character-level noise into the input data at evaluation time. Specifically, we added a controlled amount of random noise to the input text by randomly performing character replacements, insertions, and deletions with a fixed probability (noise level). For example, a noise level of 0.05 indicates that approximately 5% of the characters in the input are altered. The performance under varying noise levels is shown in the Table 7.

MeMoTune consistently outperforms LoftQ across all noise levels. Notably, it even surpasses LoftQ's clean performance at 1% noise, demonstrating improved robustness to input perturbations.

6 Conclusion

We propose MeMoTune, a measure and moment-driven fine-tuning framework for quantized LLMs, combining PEFT to adapt quantized LLMs for downstream tasks with limited memory. We introduce Gaussian-distributed parameters in LoRA to enhance results. We analyze training distribution theoretically and propose scaled probability measure gradients. Experiments show MeMoTune surpasses existing models in NLP tasks and is ideal for lightweight devices.

7 Limitations

Despite the improvements introduced by the MeMoTune in mitigating quantization-related performance loss, several limitations must be acknowledged. First, our method is validated on the English database, and its effectiveness on non-English datasets or multilingual tasks has not been fully explored, which may limit its applicability to broader language models. Moreover, the method's effectiveness has primarily been validated on popular datasets, such as Alpaca, Hh-rlhf, GSM8K, DialogSum, BillSum, and GLUE benchmark. However, its ability to address the challenges posed by more niche or broader datasets remains somewhat less explored.

Acknowledgments

This research work is supported by the Agency for Science, Technology and Research (A*STAR) under its MTC Programmatic Funds (Grant No. M23L7b0021), and by the National Natural Science Foundation of China (Grant Nos. 62272093, 62137001, and 62461146205)

References

AI@Meta. 2024. Llama 3 model card.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Ron Banner, Yury Nahshan, and Daniel Soudry. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1613–1622. JMLR.org.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting parameter-efficient tuning: Are we really there yet? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2612–2626, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit optimizers via block-wise quantization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

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

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Ashkan Jasour. 2019. Lecture 5: Duality of sos and moment based sdps. In *MIT 16.S498: Risk Aware and Robust Nonlinear Planning*. Cambridge MA. MIT OpenCourseWare.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Anastassia Kornilova and Vladimir Eidelman. 2019. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56, Hong Kong, China. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2024. Loftq: Lora-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Jiawei Liu, Lin Niu, Zhihang Yuan, Dawei Yang, Xinggang Wang, and Wenyu Liu. 2023a. Pd-quant: Post-training quantization based on prediction difference metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24427–24437.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023b. Gpt understands, too. *AI Open*.

Yijiang Liu, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. 2023c. Noisyquant: Noisy bias-enhanced post-training activation quantization for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20321–20330.

Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. 2024. SPP: sparsity-preserved parameter-efficient fine-tuning for large language models. In *ICML*. OpenReview.net.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *ACL (1)*, pages 11662–11675. Association for Computational Linguistics.

Mahdi Nikdan, Soroush Tabesh, Elvir Crncevic, and Dan Alistarh. 2024. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. In *ICML*. OpenReview.net.

Haotong Qin, Xudong Ma, Xingyu Zheng, Xiaoyang Li, Yang Zhang, Shouda Liu, Jie Luo, Xianglong Liu, and Michele Magno. 2024. Accurate lora-finetuning quantization of llms via information retention. *arXiv preprint arXiv:2402.05445*.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Melora: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. In *ACL (1)*, pages 3052–3064. Association for Computational Linguistics.

Juncheol Shin, Junhyuk So, Sein Park, Seungyeop Kang, Sungjoo Yoo, and Eunhyeok Park. 2023. Nipq: Noise proxy-based integrated pseudo-quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3852–3861.

Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi (Viji) Srinivasan, and Kailash Gopalakrishnan. 2020. Ultra-low precision 4-bit training of deep neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 1796–1807. Curran Associates, Inc.

Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and

Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Sheng Wang, Boyang Xue, Jiacheng Ye, Jiyue Jiang, Liheng Chen, Lingpeng Kong, and Chuan Wu. 2024. Prolora: Partial rotation empowers more parameter-efficient lora. In *ACL (1)*, pages 2829–2841. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *ICLR*. OpenReview.net.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1648–1665, Singapore. Association for Computational Linguistics.

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

Jing Xu and Jingzhao Zhang. 2024. Random masking finds winning tickets for parameter efficient fine-tuning. In *ICML*. OpenReview.net.

Adam X. Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. 2024a. Bayesian low-rank adaptation for large language models. In *The Twelfth International Conference on Learning Representations*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024b. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Haoran Yang, Yumeng Zhang, Jiaqi Xu, Hongyuan Lu, Pheng-Ann Heng, and Wai Lam. 2024c. Unveiling the generalization power of fine-tuned large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 884–899, Mexico City, Mexico. Association for Computational Linguistics.

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

Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. 2024. Gradient-based parameter selection for efficient fine-tuning. In *CVPR*, pages 28566–28577. IEEE.

Ke Zhu, Yin-Yin He, and Jianxin Wu. 2023. Quantized feature distillation for network quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11452–11460.

A The k-bit Quantile Quantization

For dividing the original data into 2^{k+1} intervals, we first estimate the 2^{k+1} quantiles of a normal distribution $\mathcal{N}(0,1)$, its quantile function is $Q_X(\cdot)$. Then we normalize the values of k-bit data into the $[-1,1]$. The process of determining the quantiles for quantization can be described mathematically as follows:

$$q_i = \frac{1}{2}(Q_X(\frac{i}{2^k+1}) + Q_X(\frac{i+1}{2^k+1})), \quad (11)$$

where $i = 0, 1, 2, \ldots, 2^k - 1$ represent the interval, q_i represents the quantiles of each interval. Next, we normalize the input weights into the $[-1,1]$ by $\boldsymbol{W}_N = \frac{\boldsymbol{W}}{\max|\boldsymbol{W}|}$. By finding the closest q_i to each elements in $\boldsymbol{W}_N$, we get the quantized result $\boldsymbol{i}$.

During dequantization, the quantized indices $\boldsymbol{i}$ are used to retrieve the corresponding quantiles $\boldsymbol{q}_i$, and the weight matrix is reconstructed through denormalization as:

$$\boldsymbol{W}_q = \boldsymbol{q}_i \max|\boldsymbol{W}|. \quad (12)$$

B Proofs of Theorem 1

This is a proof of the proposed gradient of the adapter's distribution parameters in Theorem 1.

Proof.

$$\begin{aligned}\nabla_{\boldsymbol{\mu_A}} \mathbb{E}_{\boldsymbol{x}}[L(f,y)] &= \frac{\partial \mathbb{E}_{\boldsymbol{x}}[L(f,y)]}{\partial \boldsymbol{A}} \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{\mu_A}} \\ &= \nabla_{\boldsymbol{A}} \mathbb{E}_{\boldsymbol{x}}[L(f,y)].\end{aligned} \quad (13)$$

$$\begin{aligned}\nabla_{\sqrt{\boldsymbol{\Sigma_{A_r}}}} \mathbb{E}_{\boldsymbol{x}}[L(f,y)] &= \frac{\partial \mathbb{E}_{\boldsymbol{x}}[L(f,y)]}{\partial \boldsymbol{A}} \frac{\partial \boldsymbol{A}}{\partial \sqrt{\boldsymbol{\Sigma_{A_r}}}} \\ &= \boldsymbol{r} \nabla_{\boldsymbol{A}} \mathbb{E}_{\boldsymbol{x}}[L(f,y)].\end{aligned} \quad (14)$$

□

C Details of Experimental Settings

C.1 Base Models

We fine-tuned the following base models to verify that MeMoTune maintains stable and excellent performance across different scenarios.

- **Llama** (7B, 13B) (Touvron et al., 2023a) and **Llama2** (7B, 13B) (Touvron et al., 2023b): Optimized for tasks such as language modeling, text generation, and reasoning, demonstrating strong capabilities in handling complex language patterns efficiently.
- **T5** (small) (Chung et al., 2024): A versatile model that frames all NLP tasks as text-to-text transformations, excelling in applications like translation, summarization, and question answering.
- **BART** (base, large) (Lewis et al., 2019): Combines bidirectional and autoregressive transformers, achieving high performance in text generation tasks such as image description, summarization, and machine translation.
- **DeBERTaV3** (base) (He et al., 2021a,b): Enhances BERT's architecture with disentangled attention mechanisms to improve contextual understanding, making it particularly effective for text classification and sentiment analysis tasks.

C.2 Dataset and Metric

NLG. NLG tasks cover a wide range, and we select three datasets with different purposes for fine-tuning for more comprehensive evaluation.

- **Alpaca** (Taori et al., 2023): Contains 52,000 instructions and demonstrations, helping to fine-tune models to understand and follow complex instructions accurately.
- **Hh-rlhf** (Bai et al., 2022): Provides Human Preference and Red Teaming Data, focusing on fine-tuning models for dialogue generation and alignment with human preferences.
- **GSM8K** (Cobbe et al., 2021): Includes 8.5K math word problems designed for reasoning tasks, such as solving grade-school math problems.

Evaluation on Alpaca and Hh-rlhf is conducted using perplexity, where lower values indicate better performance.

Summarization. We consider two datasets:

- **DialogSum** (Chen et al., 2021): A dialogue summarization dataset containing 13,460 dialogues.
- **BillSum** (Kornilova and Eidelman, 2019): A dataset containing summaries of U.S. Congressional and California state bills.

Regarding the evaluation metrics, ROUGE-1 measures unigram (word-level) overlap between the generated and reference summaries, while ROUGE-2 captures bigram overlap. ROUGE-L, based on the longest common subsequence, accounts for sentence-level structural similarity by identifying the longest co-occurring n-gram. Higher scores indicate better performance.

NLU. We use two benchmarks:

- **GLUE Benchmark** (Wang et al., 2018): General Language Understanding Evaluation Benchmark, introduced in 2018, evaluates a model's general language understanding across nine NLP tasks, including sentence similarity, inference, and sentiment analysis. The GLUE tasks used in this research are:
  - Corpus of Linguistic Acceptability (CoLA) – Assesses whether a sentence is grammatically correct, evaluated using Matthew's Correlation Coefficient (Matt).
  - Microsoft Research Paraphrase Corpus (MRPC) – Determines semantic equivalence between sentence pairs, measured by accuracy and F1 score (Acc/f1).
  - Semantic Textual Similarity Benchmark (STS-B) – Measures sentence similarity on a continuous scale, evaluated using Pearson/Spearman correlation against human judgments (P/S Corr).
- **MMLU** (Hendrycks et al., 2021): Massive Multitask Language Understanding, a multi-task benchmark covering 57 subjects across humanities, STEM, and social sciences. Performance is measured by the average accuracy across all tasks.

Higher values indicate better performance for all metrics.

C.3 Experimental Configurations

All experiments are conducted on NVIDIA A100 SXM4 GPUs.

NLG Tasks. For training Llama and Llama2 on the Alpaca and Hh-rlhf datasets, we use a cosine decay learning rate schedule with an initial learning rate of 0.01. On GSM8K, the learning rate is set to 3×10^{-4}. Fine-tuning is performed for 20,000 steps on Alpaca, 100,000 steps on Hh-rlhf, and 1,880 steps on GSM8K, with a batch size of 2 for both training and evaluation. For all NLG tasks, the scale size s is fixed at 400 for Llama and Llama2, and 100 for Mistral and Qwen2.5. The K-value is set to 2 for Llama-7B and Llama2-7B, and to 1 for Llama-13B, Llama2-13B, Mistral and Qwen2.5.

Quant	Dataset	Rank	Learning Rate	Scale	K
NF2	cola	16	1.00E-04	1000	1
NF2	cola	32	5.00E-05	600	1
NF2	mrpc	16	1.00E-04	500	1
NF2	mrpc	32	1.00E-04	600	1
NF2	stsb	16	1.00E-04	700	1
NF2	stsb	32	1.00E-04	700	1
Unif	cola	16	1.00E-04	1000	2
Unif	cola	32	1.00E-04	4000	1
Unif	mrpc	16	1.00E-04	700	1
Unif	mrpc	32	1.00E-04	100	1
Unif	stsb	16	5.00E-05	300	1
Unif	stsb	32	5.00E-05	300	1

Table 8: Hyperparameter configurations for fine-tuning 2-bit DeBERTaV3-base on NLU Tasks.

Quant	Dataset	Rank	Learning Rate	Scale	K
NF2	cola	16	1.00E-04	400	1
NF2	cola	32	5.00E-05	400	1
NF2	mrpc	16	1.00E-04	600	1
NF2	mrpc	32	1.00E-04	200	1
NF2	stsb	16	1.00E-04	1000	1
NF2	stsb	32	1.00E-04	900	2
Unif	cola	16	1.00E-04	400	2
Unif	cola	32	1.00E-04	4000	1
Unif	mrpc	16	1.00E-04	700	1
Unif	mrpc	32	1.00E-04	100	1
Unif	stsb	16	5.00E-05	300	1
Unif	stsb	32	5.00E-05	400	1

Table 9: Hyperparameter configurations for fine-tuning 4-bit DeBERTaV3-base on NLU Tasks.

Summarization Tasks. The learning rate is configured to 2×10^{-5}. Fine-tuning is conducted for 40,000 steps on the BillSum dataset and 20,000 steps on the DialogSum dataset, using a consistent batch size of 8 for both training and evaluation. The hyperparameter K is set to 4, and the scale size s is set to 1000.

NLU Tasks. Training and evaluation are conducted with a batch size of 16 and 60 epochs across all experiments. Hyperparameters, including learning rate, scale, and K vary across quantization method, datasets and rank. The detailed configurations are summarized in Table 8 (for 2-bit quantization) and Table 9 (for 4-bit quantization).

D Additional Results

D.1 NLG Task with Adapter Ranks 32

Table 10 presents the results of fine-tuning MeMoTune on the Alpaca dataset with an adapter rank of 32. Compared to the results with rank 16 in Table 2, increasing the rank to 32 leads to a performance drop for both MeMoTune and the baselines when fine-tuning 13B models, while fine-tuning 7B models shows improved performance. This highlights the importance of selecting an appropriate adapter rank based on model size and specific scenarios. Notably, MeMoTune consistently outperforms all baselines across different settings, demonstrating robust performance in various conditions.

Method	Llama		Llama2	
	7B	13B	7B	13B
LoRA	2.76	2.68	2.74	2.60
QLoRA	2.80	2.72	2.75	2.68
IR-QLoRA	2.91	2.83	2.85	2.78
LoftQ	2.75	2.66	2.71	2.59
MeMoTune	**2.70**	**2.63**	**2.66**	**2.57**

Table 10: Results of Llama and Llama2 on Alpaca. The adapters' rank is set to 32.

D.2 4-Bit DeBERTaV3-base

Table 11 shows the 4-bit fine-tuning results of DeBERTaV3-base on the GLUE tasks, where our method consistently outperforms LoftQ.

Quant	Rank	Method	CoLA Matt	MRPC Acc/f1	SST P/S Corr
NF	16	LoftQ	69.6	90.2/92.9	90.8/90.8
		MeMoTune	**70.2**	**90.4/93.1**	**90.9/90.9**
	32	LoftQ	67.2	89.2/92.3	91.0/91.2
		MeMoTune	**70.8**	**89.5/92.4**	**91.2/91.2**
Unif	16	LoftQ	67.7	88.7/92.0	88.9/88.8
		MeMoTune	**70.1**	**89.0/92.1**	**90.2/ 90.2**
	32	LoftQ	65.4	86.8/90.6	90.5/90.5
		MeMoTune	**68.8**	**89.7/92.7**	**90.7/90.7**

Table 11: Results of 4-bit DeBERTaV3-base on GLUE.

D.3 NLU Task with Adapter Ranks 32

Table 12 presents the MMLU results for the NLU task with an adapter rank of 32. Notably, MeMoTune consistently achieves the best performance, except in the case of fine-tuning Llama-7B, where its result is slightly inferior to that of LoRA. This highlights LoRA's advantage as a full-precision model. Moreover, compared to rank 16 (Table 6), fine-tuning 7B models with rank 32 generally yields better results, aligning with the conclusions in Appendix D.1. These findings suggest that the optimal rank setting may vary depending on specific scenarios.

Method	Llama		Llama2	
	7B	13B	7B	13B
LoRA	**41.5**	48.4	47.1	56.7
QLoRA	40.5	48.4	45.7	55.5
LoftQ	39.3	48.8	47.0	55.4
IR-QLoRA	39.0	47.6	46.5	54.6
MeMoTune	41.3	**49.4**	**47.7**	**56.8**

Table 12: MMLU results of Llama and Llama2 on Alpaca. The adapters' rank is set to 32.