GIFT-SW: Gaussian noise Injected Fine-Tuning of Salient Weights for LLMs

Maxim Zhelnin♣ [1], Viktor Moskvoretskii♣ [1,3], Egor Shvetsov[1], Egor Venediktov[1], Maria Krylova, Aleksandr Zuev, Evgeny Burnaev [1,2]

[1] Skolkovo Institute of Science and Technology
[2] Artificial Intelligence Research Institute
[3] HSE University

Correspondence: m.zhelnin@skol.tech ♣ indicates equal contribution.

Abstract

Parameter Efficient Fine-Tuning (PEFT) methods have gained popularity and democratized the usage of Large Language Models (LLMs). Recent studies have shown that a small subset of weights significantly impacts performance. Based on this observation, we introduce a novel PEFT method, called Gaussian noise Injected Fine Tuning of Salient Weights (GIFT-SW). Our method updates only salient columns, while injecting Gaussian noise into non-salient ones. To identify these columns, we developed a generalized sensitivity metric that extends and unifies metrics from previous studies. Experiments with LLaMA models demonstrate that GIFT-SW outperforms full fine-tuning and modern PEFT methods under the same computational budget. Moreover, GIFT-SW offers practical advantages to recover performance of models subjected to mixed-precision quantization with keeping salient weights in full precision.

1 Introduction

Modern LLMs demonstrate remarkable generalization capabilities on unseen tasks. However, fine-tuning remains crucial to enhance these models performance or to restore the performance after compression techniques like quantization (Dettmers et al., 2024; Moskvoretskii et al., 2024), pruning (Frantar and Alistarh, 2023; Kim et al., 2023), or tensor decomposition have been applied. Given the large scale of modern LLMs, fine-tuning all parameters can be computationally and memory-intensive. To overcome this challenge, Parameter Efficient Fine-Tuning schemes have been developed, aimed to improve model performance while using limited computational and memory resources.

To date, PEFT methods have not matched the accuracy of full fine-tuning (Nikdan et al., 2024), highlighting the need for new approaches that can close this gap while still minimizing resource use. Additionally, most PEFT methods involve adding extra parameters, which increases computational demands.

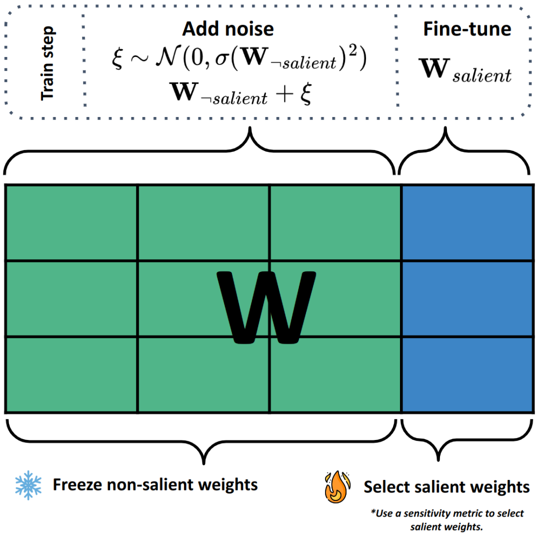
Train step
Add noise
$\xi \sim \mathcal{N}(0, \sigma(\mathbf{W}_{\neg salient})^2)$
$\mathbf{W}_{\neg salient} + \xi$
Fine-tune
$\mathbf{W}_{salient}$
W
Freeze non-salient weights
Select salient weights
*Use a sensitivity metric to select salient weights.

Figure 1: GIFT-SW procedure follows Equation 2. We first sample some noise, relative to quantization levels, then, perform forward pass, and then update salient weights only. In GIFT-SW, quantization, pruning or tensor decomposition can be applied to non-salient weights and then, salient weights can be fine-tuned effectively without changing non-salient weights structure. In our experiments we select only 128 columns of salient weights, unless specified otherwise.

To address those issues and enhance the performance of efficiently trained LLMs, we introduce a novel PEFT method, GIFT-SW. This approach focuses on updating a small subset of salient weights while injecting noise into the non-salient weights. The development of this method is grounded in observations from previous studies and the related questions they raise, which we aim to answer:

Previous research has shown that there is a small subset of salient weights which can significantly affect the effectiveness of post-training quantization (PTQ) (Dettmers et al., 2022, 2023; Kim et al.,

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6463–6480
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

2023) and pruning techniques (Yin et al., 2023; Frantar and Alistarh, 2023; Sun et al., 2023). Moreover, Gurnee et al. identified a group of "universal neurons" that are critical to a model's functionality, emphasizing the importance of selecting and updating these salient weights.

Question 1: Does updating a small subset of salient weights is sufficient to adjust the model?

Recent studies have demonstrated that Perturbed Gradient Descent (PGD), with noise injections applied both before and after the gradient step, can stabilize convergence and help prevent overfitting (Poole et al., 2014; Zhu et al., 2018; Jin et al., 2021).
Question 2: Does Injecting Noise helps convergence?

PGD is commonly employed to enhance model robustness by approximating the quantization process (Shvetsov et al., 2022; Shin et al., 2023; Défossez et al., 2021). This increased robustness can aid in maintaining the quality of the quantized model.
Question 3: Does injecting noise helps robustness?

Selecting salient weights is a significant challenge, particularly in quantization and pruning, and it is central to our method. In our paper, we derive a general formulation for all previously established saliency metrics and present experiments to compare their effectiveness.

The main contributions of our work can be summarized as follows:

- We introduce a novel PEFT method for pre-trained and quantized LLMs, called GIFT-SW. It is designed to fine-tune weights in salient columns while injecting Gaussian noise into non-salient weights, which are kept frozen during training.

- We generalize sensitivity metrics for identifying salient columns in pre-trained LLMs. We compare various novel and existing instances of the proposed general form and identify a new metric, which on average outperform previously studied in the literature metrics(Xiao et al., 2023; Lee et al., 2024).

- Experiments demonstrate that GIFT-SW outperforms modern PEFT methods and full fine-tuning baselines across most zero-shot tasks. GIFT-SW for LLaMA models achieve comparable accuracy to the corresponding state-of-the-art TÜLU2 models, despite fine-tuning only 3% of the parameters and utilizing ten times less computational resources.

We provide the code with GIFT-SW integrated into the popular PEFT library (Mangrulkar et al., 2022), making it easy to use [1].

2 Related Work

2.1 Parameter efficient fine-tuning of LLM

One of the most popular method with high efficiency is LoRA (Hu et al., 2021), which trains the low-rank adapters. Recent modifications to the method aim to improve the initialization of the adapters (Liu et al., 2024) and enhance the low-rank representation of pre-trained weights by adding sparse adapters (Nikdan et al., 2024). Another improvement of the learning capacity of LoRA is given by DoRA (Liu et al., 2024), which fine-tunes magnitude and direction components of the pre-trained weights. This method achieves considerable performance across various fine-tuning tasks.

2.2 Salient Weights in LLMs

The identification of salient weights[2] is one of the main problems in weight pruning. Recently, several approaches have been proposed to identify such weights in LLMs, including SparseGPT (Frantar and Alistarh, 2023), Wanda (Sun et al., 2023), and OWL (Yin et al., 2023).

Dettmers et al.'s (2022) demonstrated that a small subset of outliers in input activations has a substantial impact on LLM performance, highlighting the relationship between the activation outliers and the salient weights. Many subsequent Post-Training Quantization (PTQ) methods used similar or identical pruning metrics to identify these salient weights (Dettmers et al., 2023; Xiao et al., 2023; Lee et al., 2024).

In our work, we generalize the identification metrics for salient weights by considering metrics from both the literature on pruning and quantization.

2.3 Structured and Non-structured Salient Weights selection

Since salient weights represent only a small percentage of all weights, a simple approach to preserve them is storing them in a sparse matrix. Dettmers et al. (2023) showed this method is computationally efficient and enhances performance.

[1] `https://github.com/On-Point-RND/GIFT_SW`

[2] In our work, we use the terms **salient weights** and weight **outliers** interchangeably.

Meanwhile, Xiao et al. (2023) found that activation outliers are limited to a few weight channels, which SmoothQuant addresses by identifying outlier columns with a small calibration dataset. This idea is expanded in QUIK (Ashkboos et al., 2023), where outlier columns are kept at full precision while others are quantized using GPTQ (Frantar et al., 2022). OWQ (Lee et al., 2024) follows a similar approach but utilizes an OBD-based metric (LeCun et al., 1989).

Given the lack of literature on whether structured or unstructured salient weight selection yields better results, and motivated by the computational efficiency noted in (Ashkboos et al., 2023), we adopt structured column-wise salient weight selection in our work.

2.4 Noise Injections

In this section, we briefly describe Gaussian Noise Injections (GNI) and its benefits. Then we discuss studies which show close similarity between quantization noise and Gaussian Noise. Therefore, to study our third question, we sample noise relative to quantization levels, leaving other sampling options for future work.

Gaussian Noise Injections (GNI). Perturbed Gradient Descent (PGD) is a family of methods that involve adding or multiplying weights with samples from some random distribution, during an optimization procedure. Gaussian noise injection (GNI) after the gradient step helps to escape saddle points efficiently in non-convex optimization (Jin et al., 2021). However, Gaussian noise injections before the gradient step helps to escape from the spurious local optimum (Zhu et al., 2018).

Quantization Noise Injections (QNI). Quantization aware training (QAT) is applied to mitigate accuracy degradation after quantization. However, uniform quantization [3] Q is a non-differentiable operation. For simplicity, it can be expressed as a composition of scaling and rounding operations, $Q(\mathbf{W}) = \Delta \left\lfloor \frac{\mathbf{W}}{\Delta} \right\rceil$. In terms of QAT operation Q can be efficiently approximated with quantization noise ξ such that $\xi = Q(\mathbf{W}) - \mathbf{W}$ Défossez et al. (2021); Shvetsov et al. (2022); Shin et al. (2023). Thus, training models with QNI is exactly the same as employing PGD with GNI before evaluating the gradient.

Under some assumptions the noise ξ induced by uniform quantization can often be modeled by an additive noise that is uniformly distributed (Section H), uncorrelated with the input signal, and has a white spectrum (Widrow et al., 1996). However in practice, the conditions are often not satisfied. Therefore employing Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ for ξ typically yields improved outcomes (Défossez et al., 2021; Shvetsov et al., 2022).

2.5 Straight Through Estimator

The most popular QAT technique incorporating quantization operation into the training process is Straight Through Estimation (STE) (Bengio et al., 2013; Shang et al., 2023), which basically reparameterizes gradients. However, Défossez et al.; Shin et al.'s (2021; 2023) demonstrated that STE has some disadvantages compared with QNI. More technical details about STE are provided in Section C.

3 Method

GIFT-SW consists of the following steps:

(1) Identify a fixed number of salient columns using a chosen sensitive metric, based on a small calibration set. This number remains consistent across all layers.

(2) Split columns of the matrices into subsets of salient columns and regular ones.

(3) During training, add noise to the weights in non-salient columns and update weights only in the salient columns.

Thus, the method depends on two main design choices: 1) how to choose salient columns and 2) the parameters of noise injections. We cover the choice of metrics in Section 3.1. Noise injection details are provided in Section 3.2.

GIFT-SW enables efficient inference when combined with mixed-precision quantization, outperforming low-rank PEFT methods like LoRA and DoRA by avoiding additional adapter parameters. In 4-bit settings, it achieves up to 2.5× speedup over float16 and matches the efficiency of state-of-the-art quantization methods.[4]

3.1 Generalizing parameter sensitivity metrics

Several approaches have been proposed recently to identify weights sensitive to quantization (Dettmers

[3] For the reader not familiar with uniform quantization, we discuss it in more details in Section A.

[4] Efficient implementation is available with CUDA kernels from QUIK (Ashkboos et al., 2023) and OWQ (Lee et al., 2024).

	LLaMA2-7b		LLaMA2-13b		LLaMA3-8b	
	TÜLU-V2-mix	OpenOrca	TÜLU-V2-mix	OpenOrca	TÜLU-V2-mix	OpenOrca
FT	71.97	71.88	75.09	75.21	76.13	**77.02**
LoRA	71.78	70.89	74.03	74.01	75.91	75.63
DoRA	72.03	70.97	73.97	73.96	75.89	75.72
GIFT-SW	**73.33**	**72.33**	**75.93**	**76.02**	**76.37**	76.78

Table 1: Mean accuracy of LLaMA models fine-tuned with various instructive datasets and different methods.

Bits	**Method**	**LLaMA2-7b**	**LLaMA2-13b**	**LLaMA3-8b**
4 bit	STE	72.43	**75.29**	74.84
	QUIK + LORA	63.99	71.08	74.27
	GIFT-SW	**72.53**	74.50	**75.46**
3 bit	STE	69.82	**74.37**	70.24
	QUIK + LORA	62.91	71.30	71.65
	GIFT-SW	**71.00**	74.34	**73.27**
2 bit	STE	58.20	62.19	48.96
	QUIK + LORA	41.44	47.14	53.80
	GIFT-SW	**61.09**	**67.61**	**58.89**

Table 2: Mean accuracy of quantized and then fine-tuned models. For fine-tuning we used TÜLU-V2-mix.

et al., 2023) or pruning (Sun et al., 2023). We generalize them as metrics for sensitivity to perturbations, and by applying these metrics, we determine which columns are more susceptible to degradation. Therefore, we avoid adding noise to such columns and use them to fine-tune the model.

General Sensitivity Metric is written for a column j of weight matrix $\mathbf{W}$ as

$$s_j = \|\mathbf{D}_j\|_\tau \|\mathbf{X}_j\|_\rho^\gamma, \qquad (1)$$

where $\mathbf{D}_j$ is a measure of weights perturbation, s_j denotes sensitivity of the column to perturbations, $\mathbf{X}$ is the input feature, and γ takes on one of the following values $1/2, 1, 2$. As discussed in Section 2.4 we could apply GNI as a source of perturbations, then we would compute $\mathbf{D}_j = \mathbf{W}_{:,j} + \xi$. However, sampling noise ξ is not deterministic. To approximate an influence of the noise ξ we utilize perturbations caused by quantization.[5] That would lead to $\mathbf{D}_j = \mathbf{W}_{:,j} - Q(\mathbf{W}_{:,j})$, where $Q(\mathbf{W}_{:,j})$ corresponds to the weights subjected to uniform symmetric quantization (see Appendix A).

The input feature $\mathbf{X}$ for each layer is computed using a number of random sentences from a calibration dataset. After that, sensitivity values s_j are estimated for individual columns. Columns with the highest values are identified as the salient columns. Some details about the calibration dataset is described in Section 4.1.

Special Cases. The metric $\|\mathbf{X}\|_\infty$ is employed in QUIK (Ashkboos et al., 2023) and SmoothQuant (Xiao et al., 2023). OWQ (Lee et al., 2024) adopts $\lambda_j \|\mathbf{D}_j\|_2^2$, where $\lambda_j = \|\mathbf{X}_j\|_2^2$ is the j-th diagonal element of the Hessian matrix $\mathbf{H}$ for the layer quantization error. It can be seen, that the sensitivity metric used in OWQ is a modification for column quantization of the salience measure provided in OBD (LeCun et al., 1989) for network pruning. A metric proposed in Wanda (Sun et al., 2023) is element-wise variant of the metric $\|\mathbf{D}_j\|_1 \|\mathbf{X}_j\|_2$, which can be easily obtained from Equation 1 with pruning as a source of perturbations for $\mathbf{D}_j$.

Parameter Choice. In our method we choose $\gamma = 1, \rho = \infty, \tau = \infty$. In contrast to Wanda, we use l_∞ norm due to the following observations, examples contained in a calibration dataset induce different values of the input feature, a use of l_2 norm leads to averaging of the values along input channels. Therefore, the appearance of the outlier values in the input activation can be obscured by a large number of lower values. The same conclusions can be also applied to the weight error. In the case of the l_2 norm, the error for each channel includes all deviations between the quantized and

[5] Optionally, one could use weight pruning as a source of perturbations or any other.

original weights. Therefore, rare considerable errors can be mitigated by a large number of small deviations. Further ablations are performed in Section 6.

3.2 Quantization Noise Injection

To enhance our fine-tuning procedure with QNI, we avoid perturbing sensitive weights. After identifying sensitive or salient columns, we inject quantization noise only into non-salient columns across all layers, as shown in Figure 1.

The scale parameters of the Gaussian noise are determined by the quantization step sizes, which are computed for each layer prior to the training process.

For the weight matrix $\mathbf{W}$ of a given layer in the model, the process of noise injection can be described as follows. During each forward pass in the training phase, we first sample elements of noise matrix $\mathbf{\Omega}$ from standard normal distribution $\mathcal{N}(0, 1)$. Subsequently, the matrix $\mathbf{\Omega}$ is scaled with the quantization step size $\mathbf{\Delta}$. Finally, we add scaled noise to weights of non-salient columns $\mathbf{W}_{[:,\neg salient]}$. The operation of the noise injection $\mho$ is given as

$$\mho(\mathbf{W}) = \begin{cases} \mathbf{W}_{[:,salient]}, \\ \mathbf{W}_{[:,\neg salient]} + \frac{1}{2}\mathrm{diag}(\mathbf{\Delta})\mathbf{\Omega} \end{cases}, \tag{2}$$

where $\mathrm{diag}(\mathbf{\Delta})$ is the diagonal matrix with elements of the vector $\mathbf{\Delta}$.

Only weights of the salient columns $\mathbf{W}_{[:,salient]}$ are updated during training, whereas weights of other columns $\mathbf{W}_{[:,\neg salient]}$ are frozen. We do not inject noise to salient weights since small perturbations in them can cause high model degradation.

The quantization step size $\mathbf{\Delta}$ is determined only for weights in non-salient columns $\mathbf{W}_{[:,\neg salient]}$. To closer match the initial distribution of the weights, quantization scale factors including in $\mathbf{\Delta}$ are estimated for each row individually. For i-s row the scale factor Δ_i is computed as:

$$\Delta_i = \frac{\alpha_i}{2^{b-1} - 1}, \tag{3}$$

where b is the bit-width and α_i is the quantization parameter. As in quantization methods, smaller bit-width b corresponds to higher quantization noise. The parameter α_i is estimated by optimizing weight error through linear search as discussed in Appendix A.

Based on Equations 2 and 3, the variance of the injected noise is determined by the distribution of non-salient weights across rows. We exclude salient columns from this distribution, as the salient weights may induce large quantization error and distort row-wise scale factors. This approach helps us to minimize the noise variance, which, in turn, leads to a reduction in the deviation of the non-salient weights during training.

Sampling noise in this manner enables the quantization pre-training discussed in Section 6.4.

4 Experiments

In this section, we describe the experimental procedure used to test the performance of GIFT-SW compared to others. Training details could be found in Appendix D

4.1 Data

Following previous studies (Nikdan et al., 2024; Hu et al., 2021; Liu et al., 2024), we focus on the instruction tuning task. For this purpose, we use the TÜLU-V2-Mix as the main source of data (Ivison et al., 2023), as it encompasses a wide range of instructions from different sources. This dataset has been filtered, contains a substantial amount of data without being too large, and models tuned to this set show superior performance. Additionally, we utilize the OpenOrca dataset (Mukherjee et al., 2023) to demonstrate that our method does not depend on a specific set of instructions.

The sensitivity metrics to find salient columns are estimated based on 512 random sentences from the Pile validation dataset (Xiao et al., 2023).

4.2 Baselines

We consider several baselines for both full precision and quantized experiments. All baselines are applied to LLaMA2-7b, LLaMA2-13b and LLaMA3-8b.

Full precision version includes the choice of baselines, following recent studies (Liu et al., 2024; Nikdan et al., 2024). We employ:

- LoRA is a widely used adapter-based method (Hu et al., 2021)
- DoRA is modification of LoRA outperforming all current PEFT methods (Liu et al., 2024)
- FT is full fine-tuning of all parameters

We do not include PEFT methods connected with prompt tuning, as they show worse performance

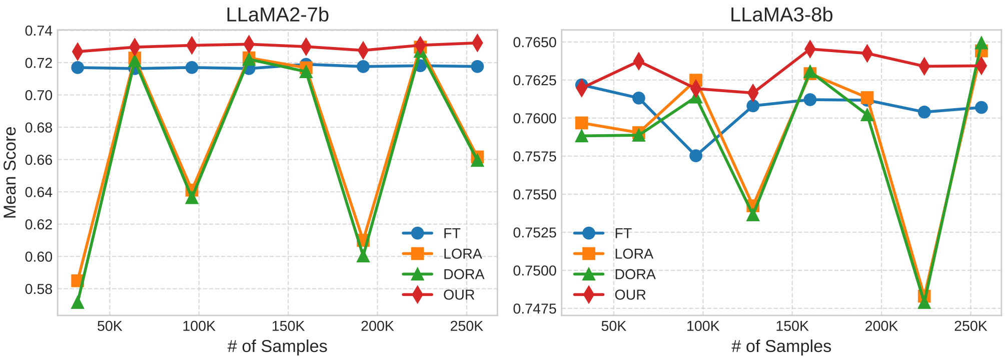

Figure 2: Mean performance of different fine-tuning approaches for LLaMA models with scaling data budget. GIFT-SW shows superior performance with nearly all data budgets, also being as stable as full fine-tuning.

compared to adapter-based methods (Xu et al., 2023).

Quantized version is presented by baselines of only weight quantization at $\{4, 3, 2\}$ bit-widths:

- STE is quantization-aware fine-tuning of all parameters of a pre-trained model (Bengio et al., 2013). During fine-tuning all parameters are trained, but 128 salient columns are updated in full-precision without quantization.
- QUIK + LoRA is an application of LoRA to the QUIK quantized model. Only low-rank adapters are trained, while the quantized weights and the salient weights are frozen.

QUIK is a mixed-precision quantization method, that leverages GPTQ for quantization non-salient columns, while keeping the salient weight in full-precision (Frantar et al., 2022; Ashkboos et al., 2023). Due to the techniques, QUIK achieves the highest performance among PTQ methods, such as GTPQ (Frantar et al., 2022), AWQ (Lin et al., 2023), SmoothQuant (Xiao et al., 2023).

4.3 Evaluation and Datasets

We perform a comprehensive evaluation measuring zero-shot performance on HellaSwag (Zellers et al., 2019), BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., 2021), PiQA (Tata and Patel, 2003), ARC-easy, and ARC-challenge (Clark et al., 2018) using the LM Eval Harness (Gao et al., 2023). The choice of baselines is similar to those in previous studies (Egiazarian et al., 2024; Frantar et al., 2022; van Baalen et al., 2024). We demonstrate average accuracy across all the datasets, detailed per-dataset comparison can be found in Section E.

	LLaMA2-7b		LLaMA2-13b	
	Performance	Compute†	Performance	Compute†
TÜLU2	73.49	6.7B / 5K	75.51	13B / 5K
TÜLU2-DPO	73.8	6.7B / 5K	75.53	13B / 11K
GIFT-SW	73.33	174M / 500	75.93	272M / 500

Table 3: Comparison of Performance and Compute for LLaMA2 Models using our fine-tuning method versus original TÜLU2 models. Note: Compute values are represented as Trainable Parameters / Iterations.

Furthermore, we assess model performance on a range of more challenging benchmarks, such as MMLU (Hendrycks et al., 2021), open-ended generation tasks in TriviaQA (Joshi et al., 2017), and the instruction-following evaluation suite IFEval (Zhou et al., 2023).

4.4 Compute Budget

In all experiments, the number of salient columns in the models is fixed at 128. Furthermore, we fix our training budget at 500 training iterations, unless specified otherwise. According to a recent study (Komatsuzaki, 2019), it is more effective to train for one epoch with a larger dataset rather than multiple epochs with less data. Therefore, all 500 iterations are performed within one epoch with no instruction repetitions.

5 Results

In this section, we present the results of our computational experiments and answer the questions posed in Section 1. In short, our results are as follows:

Q1: The results confirm that fine-tuning a subset of salient weights produces results comparable to those obtained using low-rank adapters.

Method	QA / Reasoning		IfEval			
	TriviaQA	MMLU	Prompt (A)	Inst (A)	Prompt (B)	Inst (B)
FP	52.50	40.78	18.67	31.89	21.26	34.41
full FT	**54.61**	42.54	19.41	32.01	20.70	33.21
GIFT-SW	53.41	**44.20**	**21.63**	**34.17**	**23.84**	**36.45**
LoRA	53.53	41.48	14.42	22.42	16.27	24.22
DoRA	51.88	41.54	7.02	7.43	7.58	7.79
QUIK 4bit	–	–	19.78	33.09	21.07	34.41
GIFT-SW 4bit	–	–	**21.26**	**33.69**	**22.74**	**35.61**

Table 4: Performance of LLaMA2-7b models after fine-tuning (full and PEFT) and 4-bit quantized fine-tuning on a subset of TÜLU-V2-mix. Columns show exact match on TriviaQA, average accuracy on MMLU, and IfEval prompt-/instance-level metrics on two evaluation variants (A, B).

Q2: Noise injections lead to improved model performance.

Q3: We could not confirm that models fine-tuned with noise injections are more robust to further quantization.

5.1 Full Precision

The average performance across closed-form common sense reasoning evaluation benchmarks for full precision models is presented in Table 1. GIFT-SW generally shows superior metrics across most models and instruction sets. However, we observe slight underperformance in LLaMA3 on the OpenOrca subset, where full training proves superior. This issue likely stems from the choice of learning rate and schedule, which can impact the tuning of outliers.

Table 4 reports the exact match score for TriviaQA and mean accuracy for MMLU. While GIFT-SW achieves performance comparable to LoRA on TriviaQA, it significantly outperforms both full fine-tuning and other PEFT methods on MMLU. Moreover, GIFT-SW improves effectiveness on the insturiction-following tasks.

Notably, during training of LLaMA-2-7B using a 4-GPU data parallel setup, GIFT-SW demonstrates greater memory efficiency than LoRA, requiring only 148GB of total GPU memory compared to LoRA's 156GB. This 8GB reduction is attributed to the selective fine-tuning mechanism of GIFT-SW, which improves efficiency without introducing additional parameters.

5.2 Quantized Models

Table 2 presents the average performance of LLaMA family models quantized at different precisions: 4, 3, and 2 bits. For 4 and 3 bits GIFT-SW achieves comparable quality with STE, however, latter one requires significantly more compute. In the 2-bit setting, GIFT-SW shows a substantial quality improvement, surpassing the second-ranked model by over 5 points.

In addition, Table 4 demonstrate that GIFT-SW can enhance the quality of the LLaMA2-7B model after 4-bit quantization on more challenging benchmarks, including MMLU, the open-ended TriviaQA benchmark, and instruction-following tasks.

To demonstrate the efficiency of our method for other model architectures, we also conduct experiments with Falcon-7b (Almazrouei et al., 2023) and GLM (Du et al., 2022) quantized into 3 bits. Table 6 demonstrates that GIFT-SW effectively restores quality of the models after quantization.

5.3 Comparison with TÜLU2

We compare GIFT-SW with TÜLU2 models (Ivison et al., 2023), which are LLaMA2 models full fine-tuned using instructions TÜLU-V2-Mix, and then aligned with DPO (Rafailov et al., 2023). These models are among the top-performing LLaMA2 modifications, but demand significant computational resources.

In Table 3, we show that by applying GIFT-SW with significantly lower computational budget (a smaller number of parameters and iterations) we achieve comparable results for LLaMA2-7b and outperform TÜLU2 for 13b.

5.4 Data Scaling Properties

We perform experiments to explore the performance of GIFT-SW and baselines with scaling data using LLaMA2 and LLaMA3 models. The results reported in Figure 2 show that while LoRA and DoRA exhibit unstable performance with scaling data, our method and full fine-tuning are more stable. Moreover, GIFT-SW method consistently ranks first across nearly all data budgets.

6 Ablation

6.1 Comparison sensitivity metrics

We study sensitivity metrics with respect to different noise levels (various perturbation magnitudes), which translate into varying quantization precision. In this experiment, the non-salient weights of LLaMA2 and TÜLU2 with 7B and 13B parameters. Models are quantized with QUIK, the salient weights are not updated. We select 128 columns of salient weights.

Bits	Model	$\|\mathbf{D}_j\|_2^2\|\mathbf{X}_j\|_2^2$	$\|\mathbf{D}_j\|_\infty\|\mathbf{X}_j\|_\infty$	$\|\mathbf{X}_j\|_\infty$	$\|\mathbf{D}_j\|_\infty\|\mathbf{X}_j\|_\infty^{1/2}$	$\|\mathbf{D}_j\|_\infty\|\mathbf{X}_j\|_\infty^2$
4 bit	LLaMA2-7b	**69.86**	69.85	69.68	69.55	69.52
	TÜLU2-7b	72.94	**73.17**	72.77	72.22	72.78
	LLaMA2-13b	72.92	**72.99**	72.83	72.83	72.56
	TÜLU2-13b	75.12	74.86	75.19	**75.47**	75.17
3 bit	LLaMA2-7b	67.50	**68.31**	67.47	68.09	67.86
	TÜLU2-7b	70.91	**71.30**	70.85	71.14	70.88
	LLaMA2-13b	71.92	71.59	**72.10**	71.77	71.45
	TÜLU2-13b	**74.33**	74.07	74.07	74.09	74.31
2 bit	LLaMA2-7b	45.86	46.78	45.99	46.81	**46.83**
	TÜLU2-7b	**54.84**	46.85	46.78	48.56	48.20
	LLaMA2-13b	57.07	**57.36**	51.83	57.30	56.73
	TÜLU2-13b	59.62	59.62	59.43	**60.67**	59.39

Table 5: Performance of LLaMA2 and TÜLU2 models after QUIK quantization with salient columns selected via various metrics. Weight perturbation is given by $\mathbf{D}_j = \mathbf{W}_{:,j} - Q(\mathbf{W}_{:,j})$.

Model	Method	BoolQ	HellaSwag	Winogrande	Average
Falcon-7b	FP	73.6	76.4	**70.7**	73.5
	QUIK 3 bit	65.5	73.4	67.4	68.7
	GIFT-SW 3 bit	**77.3**	**77.5**	70.2	**75.0**
GLM-10b	FP	37.8	37.6	52.5	42.6
	QUIK 3 bit	37.8	36.0	**53.1**	42.3
	GIFT-SW 3 bit	37.8	**39.4**	51.7	**43.0**

Table 6: Performance of Falcon-7b and GLM after QUIK quantization and fine-tuning with GIFT-SW using the TÜLU-V2-mix subset.

Method	4 bit	3 bit	2 bit
Salient FT	72.82	**71.06**	59.82
Pre-GIFT-SW	**73.15**	70.24	47.08
Post-GIFT-SW	72.53	71.00	**61.09**

Table 7: Mean performance for quantized models with or without applying GIFT-SW before or after quantization, results are demonstrated for LLaMA2-7b model.

Mean results for zero-shot tasks in Table 5 show that for most precisions, the best performance is achieved with salient columns identified by Equation 1 with $\gamma = 1, \rho = \infty, \tau = \infty$ (second column). Columns identified by the squared l_2 norm of the input feature (the OWQ metric) show better performance only for TÜLU2 quantized to 3 and 2 bits. Choosing salient columns solely by the input features (the QUIK metric) leads to underperformance, especially for 2 bit. Therefore, identifying salient columns sensitive to quantization noise requires considering both the weight quantization error and the maximum values of input activation.

Based on the results, we chose the best-performing sensitivity metric with $\gamma = 1, \rho = \infty, \tau = \infty$. However, the results do not reveal a clear rule for selecting the optimal sensitivity metric, as performance varies across different bit-widths and models with no discernible pattern. Further mathematical exploration of optimal sensitivity metric is presented in Appendix G.

6.2 Noise Injection Impact

To ablate the importance of QNI in the full-precision setting, we measure the mean performance of LLaMA2 models with and without noise injections for both salient columns fine-tuning and full fine-tuning. In the latter case, the noise is applied to the entire weight matrix.

The results in Table 9 show that QNI consistently enhances the performance of outlier fine-tuning. Although QNI can reduce performance when applied to the entire network, it still benefits LLaMA3-8b. Notably, outlier fine-tuning outperforms full fine-tuning, but only when QNI is used.

6.3 Noise Type Impact

Table 8 presents the results of experiments involving Gaussian and uniform noise injections during the fine-tuning of salient weights in Llama2-7b on the TULU-V2 mix instructive dataset. The results indicate that GIFT-SW slightly outperforms fine-tuning with uniform noise injection. Consequently, we adopt Gaussian noise injection for fine-tuning.

6.4 Quantization Before and After Training

From studies related to QAT, it is known that pre-training a model with noise injection enables to improve its predictive capabilities after quantiza-

Model	Orig	GIFT-SW	Uniform Noise
LLaMA2-7b	70.50	**73.33**	73.11

Table 8: Performance of LLaMA 2-7B after fine-tuning salient weights with Gaussian or uniform noise injections.

Model	Salient Weights FT		Full FT	
	w/ Noise	w/o Noise	w/ Noise	w/o Noise
LLaMA2-7b	**73.33**	73.16	71.64	71.97
LLaMA2-13b	**75.93**	74.80	74.58	75.09
LLaMA3-8b	**76.37**	75.45	76.32	76.13

Table 9: Mean Performance of LLaMA models with and without Noise Injection for salient weights fine-tuning and full model fine-tuning

tion (Défossez et al., 2021; Shvetsov et al., 2022). Based on those observations, in this section we examine the performance of the quantized LLaMA2-7b after fine-tuning full precision salient columns in several settings:

- **Pre-GIFT-SW**. Applying GIFT-SW prior to the quantization.
- **Post-GIFT-SW**. Applying GIFT-SW after the quantization.
- **Salient FT**. Fine-tuning salient columns after quantization with no noise injected

In the case of the pre-training, the bit-width for the model quantization corresponds to the noise level injected during the training. For the post-training, the noise injection is always performed at 4 bit.

Table 7 presents the average scores achieved by the models across evaluation benchmark. In the case of 4 bit quantization the Pre-GIFT-SW model considerable outperforms other models. But in the case of 3 and 2 bits, fine-tuning salient columns after quantization enables to achieve quantized models better generative capabilities.

The cause is the significant deviation of the quantized weights from original values induced by the extremely low-bit quantization. As a result, the interrelations between the salient weights and the quantized weights are disrupted, and the positive effect of pre-training disappears. However, post-training of the salient weight enables to form them new relations with other weights, so the model partially recovers its generative capabilities.

Also it can be observed that application of **Post-GIFT-SW** and **Salient FT** to model quantized in 3 bit gives the similar scores. But in the case of 2 bit quantization, the noise injection improves the fine-tuning of the quantized model.

7 Conclusion

In this paper, we introduce GIFT-SW, a parameter-efficient fine-tuning method that trains only weights in a small subset of salient columns while injecting quantization noise into the frozen weights. GIFT-SW proves to be superior to previous fine-tuning strategies in both full precision and quantized settings, requiring less compute budget. In data scaling experiments, GIFT-SW demonstrates greater stability than previous PEFT methods and outperforms both PEFT and full fine-tuning across nearly all data budgets. Our ablation studies show that QNI is beneficial but only with salient weights. Although GIFT-SW outperforms previous methods, further research is needed to determine how to maximize its performance in quantized settings.

We generalize the criterion for selecting salient columns from previous studies and empirically compare various parameters. Our experiments show that while some criteria perform better than others, none emerge as a clear dominant choice. This significant finding underscores the need for further research to refine these criteria.

8 Acknowledgment

This work was supported by the Ministry of Economic Development of the Russian Federation (code 25-139-66879-1-0003).

9 Limitations

We find the main limitations of our work as follows:

1. We report results of GIFT-SW exclusively for LLaMA models. Currently, numerous open-source pre-trained LLMs with high generative capabilities are available. However, LLaMA models are the most commonly chosen for studying the efficiency of modern PEFT and quantization methods. Despite the architectural similarities among most LLMs, future experiments with different models are necessary.

2. For quantizing models, we use only the GPTQ method, which is widely used for mixed-precision quantization of LLMs. This method improves the performance of quantized models by aggregating quantization error into columns stored in full precision. However,

GIFT-SW can be easily integrated with other methods, such as conventional RTN or QuantEase.

3. Experiments with GIFT-SW report results for salient columns selected using the sensitivity metric (1) with $\gamma = 1$, $\rho = \infty$, $\tau = \infty$. Our proposed metric, based on our analysis, shows high sensitivity of the salient columns to quantization in most LLaMA2 cases. However, other sensitivity metrics may yield better performance for GIFT-SW and mixed-precision quantization in different LLMs.

4. Noise parameters for fine-tuning the salient weights are determined using the QNI approach. However, other noise distributions may also enhance the fine-tuning process. Identifying the optimal noise distribution is beyond the scope of this paper.

5. In this study, we focus on developing the GIFT-SW algorithm for effective fine-tuning of LLMs, but we do not provide computationally efficient implementations of CUDA kernels for the algorithm. In the future, CUDA kernels for GIFT-SW can be developed based on the code from QUIK (Ashkboos et al., 2023) and OWQ (Lee et al., 2024).

6. We train GIFT-SW with only a few fine-tuning instruction sets, selected for their size and high benchmark results in previous studies. However, expanding the number of fine-tuning sets could make the experiments more comprehensive.

7. We evaluate our method using six distinct benchmarks inherited from various previous studies. In future research, it would be beneficial to include more benchmarks to gain additional insights.

10 Ethical Statement

The GIFT-SW method poses risks similar to those of any PEFT method. For example, it omits explicit safety training measures, so could be applied to fine-tune LLMs for generating harmful content. Also it can be applied to tailor LLMs to tailor highly specific and potentially dangerous outputs.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *Preprint*, arXiv:2311.16867.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. 2023. Towards end-to-end 4-bit inference on generative large language models. *arXiv preprint arXiv:2310.09259*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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

Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. 2021. Differentiable model compression via pseudo quantization noise. *arXiv preprint arXiv:2104.09987*.

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

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

Tim Dettmers, Ruslan A Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. In *The Twelfth International Conference on Learning Representations*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. *Preprint*, arXiv:2103.10360.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10323–10337.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. 2024. Universal neurons in gpt2 language models. *arXiv preprint arXiv:2401.12181*.

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

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

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *Preprint*, arXiv:2311.10702.

Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. 2021. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. *Journal of the ACM (JACM)*, 68(2):1–29.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2023. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*.

Aran Komatsuzaki. 2019. One epoch is all you need. *arXiv preprint arXiv:1906.06669*.

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

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Xiaofan Lin, Cong Zhao, and Wei Pan. 2017. Towards accurate binary convolutional neural network. *Advances in neural information processing systems*, 30.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

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`.

Viktor Moskvoretskii, Alexander Panchenko, and Irina Nikishina. 2024. Are large language models good at lexical semantics? a case of taxonomy learning. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1498–1510, Torino, Italia. ELRA and ICCL.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Mahdi Nikdan, Soroush Tabesh, and Dan Alistarh. 2024. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679*.

Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. 2014. Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

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

Yuzhang Shang, Zhihang Yuan, and Zhen Dong. 2023. Pb-llm: Partially binarized large language models. In *The Twelfth International Conference on Learning Representations*.

William Fleetwood Sheppard. 1897. On the calculation of the most probable values of frequency-constants, for data arranged according to equidistant division of a scale. *Proceedings of the London Mathematical Society*, 1(1):353–380.

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.

Egor Shvetsov, Dmitry Osin, Alexey Zaytsev, Ivan Koryakovskiy, Valentin Buchnev, Ilya Trofimov, and Evgeny Burnaev. 2022. Quantnas for super resolution: searching for efficient quantization-friendly architectures against quantization noise. *arXiv preprint arXiv:2208.14839*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Sandeep Tata and Jignesh M Patel. 2003. Piqa: An algebra for querying protein data sets. In *15th International Conference on Scientific and Statistical Database Management, 2003.*, pages 141–150. IEEE.

Mart van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. 2024. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*.

Stephen B Vardeman. 2005. Sheppard's correction for variances and the" quantization noise model". *IEEE Transactions on Instrumentation and Measurement*, 54(5):2117–2119.

Bernard Widrow. 1956. A study of rough amplitude quantization by means of nyquist sampling theory. *IRE Transactions on Circuit Theory*, 3(4):266–276.

Bernard Widrow and István Kollár. 2008. *Quantization noise: roundoff error in digital computation, signal processing, control, and communications*. Cambridge University Press.

Bernard Widrow, Istvan Kollar, and Ming-Chang Liu. 1996. Statistical theory of quantization. *IEEE Transactions on instrumentation and measurement*, 45(2):353–361.

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.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. 2023. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *Conference on Parsimony and Learning (Recent Spotlight Track)*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. 2018. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. *arXiv preprint arXiv:1803.00195*.

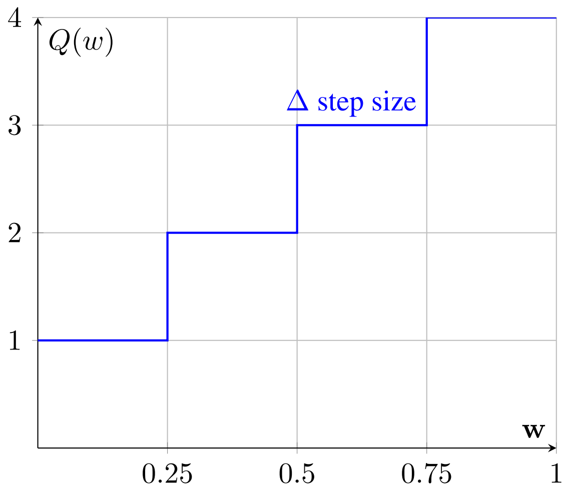

Figure 3: Uniform quantization step function with real valued one dimensional w and integer valued $Q(w)$.

A Uniform quantization

While non-uniform quantization may lead to higher compression rates, in our work we focus on uniform quantization since it widely used in efficient PTQ methods such as GPTQ, QUIK, OWQ (Frantar et al., 2022; Ashkboos et al., 2023; Lee et al., 2024). Quantization is a mapping that converts a range of full-precision values into a discrete range of values allowing usage of integer arithmetic and reduced memory consumption. For example, Fig. 3 depicts a mapping with the quantization scale size $\Delta = \frac{1}{4}$ of float values from the interval $(0, 1)$ into integer values.

In our work we apply uniform symmetric quantization with the row-wise quantization step size $\mathbf{\Delta}$. In this case, computations of quantization, dequantization and estimation of $\mathbf{\Delta}$ are performed for the bit-width b as below

$$q_{\text{min}} = -2^{b-1}, \quad q_{\text{max}} = 2^{b-1} - 1 \tag{4}$$

$$\text{clamp}(x; q_{\text{min}}, q_{\text{max}}) = \max(q_{\text{min}}, \min(x, q_{\text{max}})) \tag{5}$$

$$\mathbf{\Delta} = (\Delta_1, \dots, \Delta_n)^{\text{T}}, \quad \Delta_i = \frac{\alpha_i}{q_{\text{max}}} \tag{6}$$

$$\mathbf{W}^{\text{int}}_{i,:} = \text{clamp}\left(\left\lfloor \frac{\mathbf{W}_{i,:}}{\Delta_i} \right\rceil ; q_{\text{min}}, q_{\text{max}} \right) \tag{7}$$

$$\mathbf{W} \approx Q(\mathbf{W}) = \text{diag}(\mathbf{\Delta})\mathbf{W}^{\text{int}} \tag{8}$$

where Δ_i is the scale factor for i row $\mathbf{W}_{i,:}$, $\mathbf{W}^{int}$ denotes the matrix of the quantized weights, $\text{diag}(\mathbf{\Delta})$ is the diagonal matrix with elements of the vector $\mathbf{\Delta}$. For the given bit-width b, the parameter α_i is found for each row by performing linear grid search over the interval $[0, \max(\mathbf{W}_{i,:})]$, where $\max(\mathbf{W}_{i,:})$ is the maximum element of i row . The search is conducted to minimize layer-wise mean squared error between weights:

$$\text{argmin}_{\mathbf{\Delta}} \|\mathbf{W} - Q(\mathbf{W})\|_2^2, \tag{9}$$

B Details of LLMs quantization

For only weight quantization of LLaMA and TÜLU2 models models, we apply QUIK implementation of mixed-precision GPTQ method (Ashkboos et al., 2023; Frantar et al., 2022). We isolate 128 salient columns in full-precision. Non-salient columns are subjected to uniform symmetric quantization, as discussed in Appendix A. The salient columns are identified through sensitive metrics described in Section 3.1. The Hessian matrix for the GPTQ method is computed on 128 random samples of the Wikitext-2 dataset.

C Straight Through Estimator

STE can be described in two steps:

- Obtain quantized weights $Q(\mathbf{W})$ from the real-valued parameters $\mathbf{W}$ with some quantization function Q, which is usually is non differentiable.
- Compute gradients at quantized weights $Q(\mathbf{W})$ and update real valued weights $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \tau\nabla f(Q(\mathbf{W}))$

STE makes a particular choice of a quantization function to obtain the discrete weights from the real-valued weights. This approximation can be justified in some settings (Lin et al., 2017) but in general the reasons behind its effectiveness are unknown.

D Training Details

The training was performed with 4 GPUs (40 GB each) for 500 iterations. The batch size is 128 for 7b models and 64 for 13b models. For baseline methods, the learning rate was set to 3×10^{-5} for LLaMA2 models and to 1×10^{-5} for the LLaMA3 model. We experimented with different learning rates and found these to be the most beneficial for baseline methods. We used a cosine annealing scheduler with the warmup ratio of 0.03. The LoRA and DoRA alpha and dropout values were as specified in the original papers, and the rank was set to 64 to match the number of trainable parameters in our method. Thus, the number of trainable parameters is 160M for LLaMA2-7b, 250M for LLaMA2-13b, 167M for LLaMA3-8b.

For our method, the learning rate was set to 1×10^{-4} for salient columns of LLaMA2 models and to 1×10^{-5} of the LLaMA3 model. We fixed the number of salient columns at 128, such that the number of trainable parameters is 174M for LLaMA2-7b, 272M for LLaMA2-13b, and 176M for LLaMA3-8b.

In the case of full fune-tuning with the noise injection, the learning rate was set to 3×10^{-5} and 1×10^{-5} for LLaMA2 & 3 models, correspondingly.

E Detailed Benchmark Results

In this section we report detailed benchmark results for LLaMA 2 & 3 after training with different methods. Tables 10, 11 present accuracy metrics which are achieved by the full-precision models after fine-tuning on the TÜLU-V2-mix and OpenOrca subsets. Corresponding mean values are listed in Table 1. Tables present accuracy metrics which are achieved by quantized in 4, 3, 2 bits models after fine-tuning on the TÜLU-V2-mix subset. Corresponding mean values are listed in Table 2.

Model	Method	BoolQ	HellaSwag	WinoGrande	ARC-e	ARC-c	PiQA
LLaMA2-7b	FP	78.65	**76.91**	69.93	77.99	48.63	79.71
	LoRA	80.28	76.67	69.85	76.64	47.95	79.27
	DoRA	81.93	76.27	70.09	76.05	48.89	78.94
	GIFT-SW	**82.63**	76.68	**70.80**	**80.01**	**49.91**	**79.92**
LLaMA2-13b	FP	83.27	79.77	72.69	80.43	53.67	**80.69**
	LoRA	81.10	79.57	72.77	78.91	51.28	80.52
	DoRA	81.01	79.64	72.22	78.87	51.54	80.52
	GIFT-SW	**84.22**	**80.18**	**73.24**	**82.20**	**55.38**	80.36
LLaMA3-8b	FP	83.64	79.56	74.35	**82.41**	55.72	81.12
	LoRA	83.30	79.62	75.14	80.15	56.06	81.18
	DoRA	83.61	79.53	**75.45**	80.09	55.63	81.01
	GIFT-SW	**83.88**	**80.02**	75.22	80.56	**57.00**	**81.56**

Table 10: LLaMA models performance fine-tuned with TÜLU-V2-mix subset

Model	Method	BoolQ	HellaSwag	WinoGrande	ARC-e	ARC-c	PiQA
LLaMA2-7b	FT	80.03	**77.02**	69.69	**76.64**	48.72	79.16
	LoRA	78.81	76.24	68.82	75.42	46.59	**79.43**
	DoRA	78.78	76.30	68.92	75.67	46.93	79.22
	Our Best	**82.51**	76.64	**72.22**	74.71	**48.89**	79.00
LLaMA2-13b	FT	82.66	**80.30**	73.01	79.97	54.78	**80.52**
	LoRA	81.68	79.64	72.85	78.41	51.11	80.36
	DoRA	81.65	79.64	72.93	78.28	51.19	80.09
	Our Best	**85.44**	80.07	**74.03**	**79.97**	**56.48**	80.14
LLaMA3-8b	FT	**84.37**	**80.11**	**75.93**	**81.82**	**57.85**	**82.05**
	LoRA	82.84	79.76	74.19	80.30	55.54	81.12
	DoRA	82.63	79.71	75.22	80.30	55.46	81.01
	Our Best	84.34	80.10	75.53	81.06	57.76	81.88

Table 11: LLaMA models performance fine-tuned with OpenOrca

		Benchmarks					
LLaMA	Method	BoolQ	HellaSwag	WinoGrande	ARC-e	ARC-c	PiQA
2-7b 4bit	STE	80.21	**76.27**	70.01	79.63	48.55	**79.92**
	QUIK + LORA	68.96	74.85	69.85	55.30	37.20	77.80
	GIFT-SW	**82.78**	76.14	**70.48**	**79.76**	**50.00**	79.71
2-13b 4bit	STE	**84.77**	79.16	72.69	**80.76**	**53.67**	**80.69**
	QUIK + LORA	74.89	78.01	72.22	71.76	50.17	79.43
	GIFT-SW	84.65	**79.59**	**73.01**	78.37	53.50	80.52
3-8b 4bit	STE	81.59	78.55	73.88	79.76	54.27	81.01
	QUIK + LORA	82.51	77.73	**74.66**	79.04	51.62	80.03
	GIFT-SW	**83.15**	**79.05**	74.09	**80.01**	**55.20**	**81.28**
2-7b 3bit	STE	76.79	74.19	68.19	75.04	45.65	79.05
	QUIK + LORA	63.88	72.00	66.93	61.24	38.74	74.64
	GIFT-SW	**80.46**	**74.20**	**68.90**	**75.88**	**47.35**	**79.22**
2-13b 3bit	STE	83.33	78.02	71.59	**79.92**	**53.24**	**80.09**
	QUIK + LORA	82.02	76.64	70.95	71.51	48.21	78.45
	GIFT-SW	**85.44**	**78.20**	**71.90**	79.12	51.54	79.82
3-8b 3bit	STE	75.87	74.38	69.14	74.41	49.32	78.29
	QUIK + LORA	78.72	74.54	70.72	77.31	50.60	78.02
	GIFT-SW	**80.31**	**75.98**	**71.51**	**79.63**	**52.99**	**79.22**
2-7b 2bit	STE	68.47	58.90	60.62	57.66	32.17	71.38
	QUIK + LORA	62.11	26.77	49.88	29.67	26.45	53.75
	GIFT-SW	**71.90**	**64.18**	**62.59**	**61.57**	**34.90**	**71.38**
2-13b 2bit	STE	73.09	63.74	61.40	64.14	36.09	74.70
	QUIK + LORA	59.36	41.34	55.41	40.28	27.82	58.60
	GIFT-SW	**81.99**	**69.49**	**65.43**	**70.33**	**43.17**	**75.24**
3-8b 2bit	STE	60.46	43.82	54.46	44.23	27.65	63.16
	QUIK + LORA	64.68	48.55	58.25	53.32	32.17	65.83
	GIFT-SW	**74.13**	**48.92**	**58.88**	**63.17**	**37.88**	**70.35**

Table 12: Performance of quantized LLaMA models fine-tuned with TÜLU-V2-mix subset

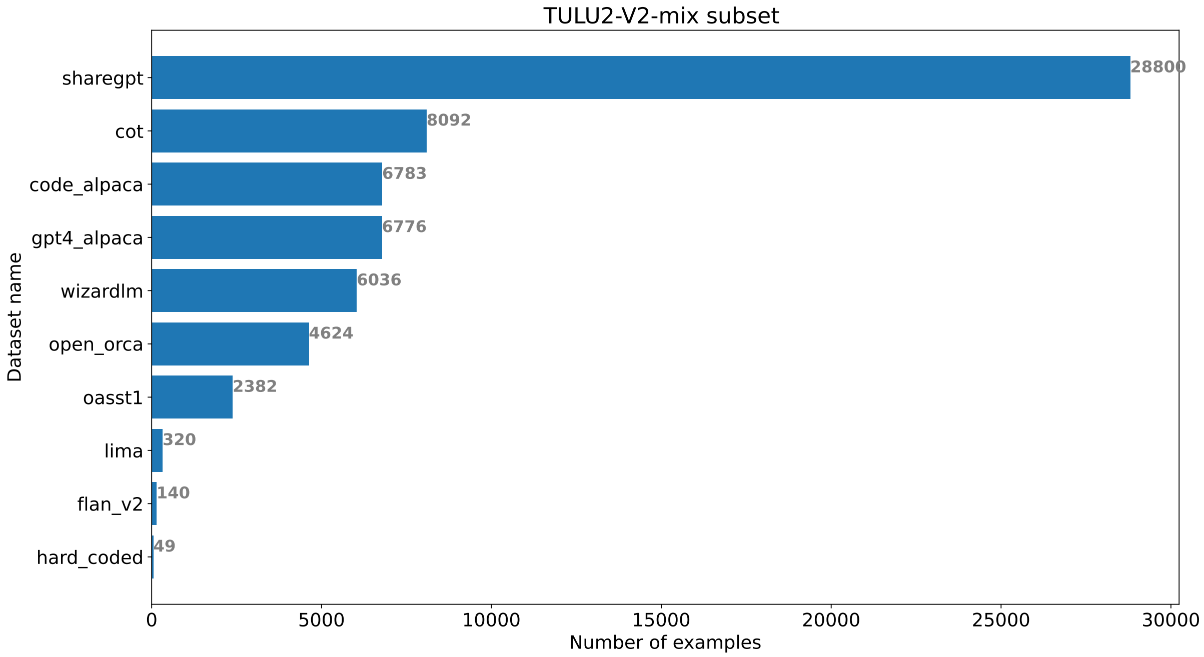

Figure 4: Number of examples in datasets included in TÜLU-V2-mix subset

F TÜLU-V2-mix subset

Figure 4 shows number of examples in datasets included in the TÜLU-V2-mix subset, which is used for fine-tuning experiments presented in this paper.

G Discussion about sensitivity metric optimality

We consider the sensitivity metric for a column j of the weight matrix $\mathbf{W}$, defined as:

$$s_j = \|\mathbf{D}_j\|_\tau \, \|\mathbf{X}_j\|_\rho^\gamma, \tag{10}$$

where:

- $\mathbf{D}_j$ is the perturbation in the weights corresponding to column j,
- $\mathbf{X}_j$ is the input feature vector,
- $\|\cdot\|_\tau$ and $\|\cdot\|_\rho$ are vector norms,
- γ is an exponent parameter.

Empirical evaluations indicate that the parameter combination $\gamma = 1$, $\tau = \infty$, and $\rho = \infty$ yields superior performance. We aim to provide a rigorous mathematical justification for this observation by analyzing the formula using Hölder's inequality and ensuring all constants are appropriately considered.

G.1 Impact of Weight Perturbations on Model Output

Consider a linear model where the output y is given by:

$$y = \mathbf{W}^\top \mathbf{X}. \tag{11}$$

A perturbation $\delta \mathbf{W}$ in the weights leads to a change in the output:

$$\delta y = (\delta \mathbf{W})^\top \mathbf{X}. \tag{12}$$

For column j, the change in the output due to perturbation $\mathbf{D}_j$ is:

$$\delta y_j = \mathbf{D}_j^\top \mathbf{X}. \tag{13}$$

Our goal is to bound $|\delta y_j|$ in terms of norms of $\mathbf{D}_j$ and $\mathbf{X}$.

First, we will apply Hölder's inequality.

Hölder's inequality states that for vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ and conjugate exponents p, q (i.e., $\frac{1}{p} + \frac{1}{q} = 1$):

$$|\mathbf{a}^\top \mathbf{b}| \leq \|\mathbf{a}\|_p \, \|\mathbf{b}\|_q. \tag{14}$$

We can use this inequality to bound $|\delta y_j|$:

$$|\delta y_j| = |\mathbf{D}_j^\top \mathbf{X}| \leq \|\mathbf{D}_j\|_p \, \|\mathbf{X}\|_q, \tag{15}$$

where p and q are conjugate exponents.

G.2 Worst Case Bound

To capture the worst-case (maximum) impact of perturbations, we aim to maximize the bound on $|\delta y_j|$. This involves selecting norms that emphasize the largest components of $\mathbf{D}_j$ and $\mathbf{X}$.

G.2.1 Choosing $p = \infty$ and $q = 1$:

- $\frac{1}{\infty} + \frac{1}{1} = 1$.
- The ℓ^∞-norm $\|\mathbf{D}_j\|_\infty$ captures the maximum absolute value in $\mathbf{D}_j$.
- The ℓ^1-norm $\|\mathbf{X}\|_1$ sums the absolute values of $\mathbf{X}$.

Using these norms:

$$|\delta y_j| \leq \|\mathbf{D}_j\|_\infty \, \|\mathbf{X}\|_1. \tag{16}$$

G.2.2 Choosing $p = 1$ and $q = \infty$:

- $\frac{1}{1} + \frac{1}{\infty} = 1$.
- The ℓ^1-norm $\|\mathbf{D}_j\|_1$ sums the absolute values of $\mathbf{D}_j$.
- The ℓ^∞-norm $\|\mathbf{X}\|_\infty$ considers the maximum absolute value in $\mathbf{X}$.

Using these norms:

$$|\delta y_j| \leq \|\mathbf{D}_j\|_1 \, \|\mathbf{X}\|_\infty. \tag{17}$$

From Equations (16) and (17), we observe that the bounds involve products of norms:

- $\|\mathbf{D}_j\|_\infty \, \|\mathbf{X}\|_1$,
- $\|\mathbf{D}_j\|_1 \, \|\mathbf{X}\|_\infty$.

However, to capture the absolute maximum effect (i.e., when both $|\delta W_{ij}|$ and $|X_i|$ are simultaneously large), the combination $\tau = \infty$ and $\rho = \infty$ is most appropriate:

$$s_j = \|\mathbf{D}_j\|_\infty \, \|\mathbf{X}_j\|_\infty^\gamma. \tag{18}$$

G.3 Optimal Scaling Choice

In our formulation, we consider only three types of exponent scaling: linear $\gamma = 1$, sublinear $\gamma = \frac{1}{2}$ and superlinear $\gamma = 2$.

The choice of linear as optimal may be motivated as it ensures the sensitivity metric scales proportionally with $\|\mathbf{X}_j\|_\infty$.

Sublinear is suboptimal as it reduces the influence of large input features and may underestimate sensitivity when large features are significant. And superlinear is suboptimal as it exaggerates the effect of large input features and may overestimate sensitivity, leading to overly conservative decisions.

H Relation between noise injection and quantization

Let us consider $\mathbf{x}$ a random variable and a quantization step Δ. Then quantization error is defined as $e = Q(x) - x$, where $Q(x) = \Delta \left\lfloor \frac{x}{\Delta} \right\rceil$, $\lfloor \cdot \rceil$ is the rounding operation. It can be written as $e = \Delta \left(\left\lfloor \frac{x}{\Delta} \right\rceil - \frac{x}{\Delta} \right)$. Thus, the quantization error e can take any value within the interval: $e \in [-\Delta/2, \Delta/2]$.

In the literature on signal processing, it is common to make the assumption that $\mathbf{x}$ follows a uniform distribution with zero-mean, $Q(x)$ and e are independent random variables (Défossez et al., 2021). In that case, $Q(x)$ follows a uniform distribution (Sheppard, 1897). Consequently, $E[e] = 0$ and Var$[e] = \Delta^2/12$. This corresponds with results in the analysis of Vardeman (2005) for approximate relationships between x and $Q(x)$.

In reality, pretrained neural network weights usually have a zero-centered normal distribution (Widrow, 1956). Therefore, it is reasonable to assume that $\mathbf{x}$ follows a normal distribution with zero-mean. Then $Q(x)$ follows a normal distribution, that implies e follows a normal distribution with $E[e] = 0$. Also, it can be shown that Var$[e] = \Delta^2/C$, where $C = \sqrt{6}$ (Widrow and Kollár, 2008).

Based on the derivation, we can write quantization operation as $Q(x) = x + e$, where e is a random variable with a normal distribution and Var$[e] = \Delta^2/C$. In our work, we used Gaussian noise with $\sigma = \Delta$, because we did not notice any difference. Moreover, it has been shown empirically that approximating quantization noise for quantization-aware training with a normal distribution with $\sigma = \Delta$ yields acceptable results, and is more preferable than using uniform noise (Défossez et al., 2021; Shvetsov et al., 2022).