Treasures in Discarded Weights for LLM Quantization

Hao Yu[1,2,3], **Yang Zhou**[3], **Bohua Chen**[3], **Zelan Yang**[3], **Shen Li**[3*], **Yong Li**[3], **Jianxin Wu**[1,2*]

[1]National Key Laboratory for Novel Software Technology, Nanjing University
[2]School of Artificial Intelligence, Nanjing University
[3]Alibaba Cloud Computing
yuh@lamda.nju.edu.cn, {xiale.zy, bohua.cbh, yangzelan.yzl, litan.ls, jiufeng.ly}@alibaba-inc.com, wujx2001@gmail.com

Abstract

In recent years, large language models (LLMs) have developed rapidly and revolutionized natural language processing. However, high storage overhead and computing costs limit LLM deployment in resource-constrained environments. Quantization algorithms can effectively compress LLMs and accelerate inference, but they lead to loss in precision, especially in low-bit scenarios. In this paper, we find that the discarded weight values caused by quantization in fact contain treasures to improve LLMs' accuracy. To excavate those hidden treasures, we construct search spaces around these discarded weights and those weights within the search space can seamlessly be incorporated into the original quantization weights. To determine which weights should be merged, we design a plug-and-play weight compensation framework to capture global information and keep the weights with the highest potential benefits. Our framework can be combined with various LLM quantization algorithms to achieve higher precision without additional inference overhead. We validate the effectiveness of our approach on widely used benchmark datasets for LLMs.

Introduction

In recent years, pre-trained large language models or LLMs (Le Scao et al. 2022; Touvron et al. 2023; Zhang et al. 2022) from the transformer (Vaswani et al. 2017) family have achieved exciting results in many complex natural language processing (NLP) tasks. LLMs have captured wide attention in both academia and industry. However, the flexibility and scalability of these models still face many limitations. Since LLMs typically have billions of parameters, they are quite expensive in terms of both computation and storage costs, which presents a huge challenge for their deployment and application.

Model quantization maps the full-precision (e.g., 32-bit floating point) weights W and activation values A to lower precision (e.g., 8-bit integer) weights W_q and activations A_q. It effectively reduces parameter counts and speeds up model inference without significantly affecting accuracy. Therefore, quantization algorithms have been widely used to deploy LLMs. Generally speaking, there are two kinds of typical quantization methods, i.e., post-training quantization (PTQ) and quantization-aware training (QAT). PTQ algorithms quantify the model's weights and activation directly. In contrast, QAT algorithms make quantization part of the model training process. In general, researchers will first perform PTQ algorithms for LLMs, and then further run QAT algorithms based on PTQ models if enough computational resources are available.

Although existing quantization methods for LLMs have demonstrated great capacity, they also have some inherent limitations. While QAT algorithms can significantly reduce accuracy degradation caused by quantization and sometimes even close to lossless, they need to retrain LLMs and require large computational resources. In contrast, PTQ algorithms are faster and more widely used. However, they often lead to a noticeable decrease in model precision. Besides, previous LLM PTQ methods (Frantar et al. 2022; Lin et al. 2024) mainly focus on adjusting weight or outlier activation layer-by-layer, and only consider the activation values of an individual layer or block. Hence, they ignore global information and may result in the accumulation of quantization errors. Both PTQ and QAT can often achieve INT8 lossless quantization for LLMs (Frantar et al. 2022; Liu et al. 2024), but further reducing to lower bits (such as INT4 and INT2) will most likely result in significant accuracy loss.

To overcome these defects of existing LLM PTQ and QAT algorithms, we propose a plug-and-play framework to improve the precision of low-bit quantization LLMs. We believe that for LLM PTQ algorithms, the discarded weights (i.e., $D = W - W_q$) still contain precious information that has not been fully mined. Therefore, we propose a novel strategy to excavate the hidden treasures in the discarded weights D. In particular, based on low-rank decomposition technique, we design various small search spaces around D. The search spaces contain multiple weights that are divisible by the quantization scales and can be incorporated into the original quantization weights seamlessly. Therefore, our framework does not add an extra inference burden. To determine which weights should be merged, we apply the global perplexity (PPL) value on the calibration set as a criterion rather than the activation change of an individual layer.

Our framework does not rely on back-propagation but can still capture global information, so only a small number of samples and hyperparameters are required and quan-

*S. Li and J. Wu are co-corresponding authors.

tization models will not overfit the calibration set. Later we will show that our framework is robust to hyperparameters and can consistently improve model accuracy with various search spaces. Our framework, namely **D**iscarded **W**eight **R**ecycling (DWR), can be combined with various existing LLM PTQ and QAT algorithms. That is, running DWR framework after LLM PTQ algorithms can further improve the precision of the quantization LLMs. Furthermore, if there are sufficient computational resources to perform QAT algorithms, then fine-tuning DWR quantization models is also better than directly fine-tuning PTQ LLMs. This indicates that DWR framework can be inserted between PTQ and QAT algorithms, thus to improve LLM quantization. We list our contributions as follows:

- To solve the problem of precision degradation caused by existing LLM low-bit quantization algorithms, we propose a unified plug-and-play framework to recycle the treasures in those discarded weight values.
- By carefully designing the search space and evaluation metric, our practical and efficient DWR framework effectively utilizes the global information, optimizes the quantization pipeline, and reduces the accuracy loss without any extra inference burden.
- A large number of experiments have verified the effectiveness of our method. In three typical LLM families, our framework can be combined with various PTQ and QAT algorithms and improve the accuracies of low-bit quantization models, which demonstrates DWR's broad applicability and effectiveness in different scenarios.

Related Work

Our work is connected to several themes in the literature, which we briefly review next.

Large Language Models (LLMs)

Large language models (Brown et al. 2020; Le Scao et al. 2022; Touvron et al. 2023; Zhang et al. 2022) are designed to understand and generate human languages. In recent years LLMs have developed rapidly and consistently show excellent performances across various NLP tasks. These breakthroughs can be attributed to their enormous scale in model size and amount of training data, as they train on massive amounts of data from different sources while containing billions or even trillions of parameters. Many LLMs have been proposed in recent years, and we will show that our framework can easily handle those models at various scales.

Post-Training Quantization (PTQ) for LLMs

PTQ algorithms quantify the parameters of LLM after the training stage, which can simply and efficiently improve the efficiency of LLM without major modifications or extensive training efforts. However, existing PTQ algorithms may introduce a certain degree of precision loss. Some LLM PTQ methods focus on quantifying weights. For example, GPT3.int8() (Dettmers et al. 2022) isolates the outlier feature dimensions into a 16-bit matrix multiplication and still multiplies most values in 8-bit. GPTQ (Frantar et al. 2022) proposes a novel group-wise quantization technique based on approximate second-order information. AWQ (Lin et al. 2024) employs an activation-aware approach by considering the significance of weight channels corresponding to larger activation magnitudes. SpQR (Dettmers et al. 2024b) identifies and separates abnormal weights, stores them with higher precision and compresses all other weights to 3–4 bits. Some PTQ works quantify both LLM's weight and activation. SmoothQuant (Xiao et al. 2023) smooths the activation outliers by offline migrating the quantization difficulty from activations to weights. OmniQuant (Shao et al. 2023) optimizes the clipping threshold to modulate the extreme values and also shifts the challenge of quantization from activations to weights. LoWC (Yao et al. 2024) applies Singular Value Decomposition or SVD (Golub and Van Loan 2013) to low-rank decompose error weight, which is similar to our idea, but it introduces additional parameters that will slow down model inference. In contrast, applying our DWR framework after finishing PTQ can dramatically improve accuracy without any additional inference burden.

Quantization-Aware Training (QAT) for LLMs

QAT algorithms enable an LLM to adapt to lower-precision fixed-point representations during training. This adaptation is designed to ensure that the fine-tuned LLM sustains its performance even after quantization to lower bit-widths. LLM-QAT (Liu et al. 2024) leverages generations produced by a pre-trained model to achieve data-free distillation, and it quantizes not only weights and activations but also key-value (KV) caches. Some QAT works only fine-tune part of the weight parameters. PEQA (Kim et al. 2024) first quantizes fully-connected (FC) layers and then fine-tunes the quantization scalar for each specific downstream task. QLoRA (Dettmers et al. 2024a) back-propagates gradients through a frozen 4-bit quantized pre-trained LLM into LoRA low-rank adapters (Hu et al. 2021). QA-LoRA (Xu et al. 2024) integrates LLM and adapters into a quantized model without accuracy loss. Generally speaking, QAT will usually be conducted after performing PTQ algorithms for LLMs. Later we will show that performing QAT after applying DWR is more effective than performing QAT directly.

Methods

Now we describe our framework. In this work, we study one of the most common quantization scenarios, i.e., weight quantization with uniform bits. We first introduce the preliminaries. Based on the discarded weights, we provide three potential search space generation methods. Then we present the DWR framework.

Preliminaries

A quantization algorithm maps a floating-point number into lower-bit integers. Suppose we have a fully-connected (FC) layer $y = Wx+b$, whose input $x \in \mathbb{R}^{n\times c}$, output $y \in \mathbb{R}^{m\times c}$ and weight $W \in \mathbb{R}^{m\times n}$. Quantization algorithms transfer the floating-point weight W into N-bit integer weight W_q with a scaling factor α and a zero-point value β. Then, the

full-precision pseudo-quantized weight $\hat{W}_q$ are

$$\hat{W}_q = \alpha(W_q - \beta). \tag{1}$$

Please note that for convenience we have omitted how to quantize W to integer weight W_q, as different LLM quantization algorithms have different strategies. We can also transfer $\hat{W}_q$ into W_q, as

$$W_q = \text{clip}(\lfloor \frac{\hat{W}_q}{\alpha} \rceil + \beta, 0, 2^N - 1), \tag{2}$$

where clip is the truncation operation and $\lfloor \cdot \rceil$ represents rounding to the nearest integer between 0 and $2^N - 1$. Therefore, information has been lost between using $\hat{W}_q$ and W, as $D = W - \hat{W}_q$—*an important reason for the accuracy loss* after quantization is with the discarded weight D! So long as D is added to $\hat{W}_q$, we can directly get the original weight W, and an LLM's accuracy is kept. But, $\hat{W}_q$ is a pseudo-quantization weight, and directly compensating D to $\hat{W}_q$ will destroy the quantization structure. Therefore, to improve the efficiency of the quantization model, we need a suitable way to integrate D and $\hat{W}_q$.

To incorporate the discarded weight D into $\hat{W}_q$ without breaking the original quantization structure, one potential solution is to add $\hat{W}_q$ and D and then divide the scale directly, i.e.,

$$\tilde{W}_q = \text{clip}(\lfloor \frac{\hat{W}_q + D}{\alpha} \rceil + \beta, 0, 2^N - 1), \tag{3}$$

where $\tilde{W}_q$ is the pseudo-quantization weight after compensation. But, this operation (proposed in the round-to-nearest or RTN method) usually performs worse than state-of-the-art quantization algorithms (Gupta et al. 2015). Considering this observation, we believe that there may be several reasons why directly merging D is at best sub-optimal, e.g.,

- It is doubtful whether directly adding the discarded weight D will bring benefits in each FC layer, and there lacks a viable criterion to evaluate it.
- This operation specifies the variable to be merge with $\hat{W}_q$ can only be D, which excludes all other potential candidates. This limits the possibility of improving LLM's accuracy.

To address these issues, we propose our DWR framework.

Generate the Search Space

To relieve the problem that there is only one candidate to merge weight, we design search spaces around the discarded weight D. Because of non-linear calculation (i.e., the clip operation) in quantization, D most likely is not the optimal solution. Instead, we *propose to generate a search space around* D, and search for the optimal merge weight. Specifically, we propose three feasible strategies.

Random Generation The first strategy is random generation. We randomly generate an orthogonal matrix $R \in \mathbb{R}^{m \times m}$, and decompose it as $R = USV^\top$, where $U \in \mathbb{R}^{m \times m}$ is an orthonormal matrix, and $S \in \mathbb{R}^{m \times m}$ is a diagonal matrix containing singular values in the decreasing order. Suppose we only use the largest k singular values. The resulting matrix R_k is an optimal approximation of R with a lower rank k, i.e., $R_k = U_k S_k V_k^\top$, where $U_k, V_k \in \mathbb{R}^{m \times k}$ contains the first k columns of U and V, and S_k is a diagonal matrix formed by the corresponding k singular values. Then we construct the search space by

$$\tilde{W}_q = \text{clip}(\lfloor \frac{\hat{W}_q + R_k D}{\alpha} \rceil + \beta, 0, 2^N - 1). \tag{4}$$

We can obtain a search space by specifying a set of different k values. Here we use orthogonal matrices because we want to avoid generating overly large values in weights.

SVD Generation A straightforward strategy is directly performing SVD on D, i.e., $D = USV^\top$. Similarly, we keep the top k singular values of S and get $D_k = U_k S_k V_k^\top$, where $U_k \in \mathbb{R}^{m \times k}$ and $V_k \in \mathbb{R}^{n \times k}$ are the top-k columns of U and V, respectively. Therefore, we have

$$\tilde{W}_q = \text{clip}(\lfloor \frac{\hat{W}_q + D_k}{\alpha} \rceil + \beta, 0, 2^N - 1). \tag{5}$$

By pre-specifying a set of different k values for searching, we obtain a feasible search space. Note that the symbols U, S, V are *defined differently* in random generation, SVD generation and the AFM generation in the next sub-section.

AFM Generation AFM is a better alternative to SVD when compressing FC layers (Yu and Wu 2023). The idea is to use PCA to low-rank compress FC layers, but AFM imitates the model outputs rather than the weight. In particular, assume $\hat{y}_q = \hat{W}_q x + b$ and $\Delta y = y - \hat{y}_q = Dx$, then AFM performs PCA on Δy. First, we compute the covariance matrix of Δy, i.e.,

$$\text{Cov}(\Delta y) = \mathbb{E}\left[\Delta y \Delta y^\top\right] - \mathbb{E}[\Delta y]\mathbb{E}[\Delta y]^\top, \tag{6}$$

where $\mathbb{E}[\cdot]$ is the expectation operator. Since $\text{Cov}(\Delta y)$ is positive semi-definite, its SVD is $\text{Cov}(\Delta y) = USU^\top$. We extract the first k columns of $U \in \mathbb{R}^{m \times m}$ into $U_k \in \mathbb{R}^{m \times k}$. Knowledge on PCA tells us

$$\Delta y - \mathbb{E}[\Delta y] \approx U_k U_k^\top (\Delta y - \mathbb{E}[\Delta y]). \tag{7}$$

This approximation is proved optimal (Wu 2020). Then, Δy can be transformed into

$$\Delta y \approx U_k U_k^\top Dx + \mathbb{E}[\Delta y] - U_k U_k^\top \mathbb{E}[\Delta y]. \tag{8}$$

Therefore, we can imitate y as

$$\begin{aligned} y &= \hat{W}_q x + b + \Delta y \\ &\approx (\hat{W}_q + U_k U_k^\top D)x + b + \mathbb{E}[\Delta y] - U_k U_k^\top \mathbb{E}[\Delta y]. \end{aligned} \tag{9}$$

Then, we can get

$$\begin{aligned} \tilde{W}_q &= \text{clip}(\lfloor \frac{\hat{W}_q + U_k U_k^\top D}{\alpha} \rceil + \beta, 0, 2^N - 1), \\ \tilde{b} &= b + \mathbb{E}[\Delta y] - U_k U_k^\top \mathbb{E}[\Delta y]. \end{aligned} \tag{10}$$

This also means that we update both $\tilde{W}_q$ and $\tilde{b}$ during the search process. Because collecting Δy during inference is

Algorithm 1: The DWR Framework

Input: The original large language model M and its low-bit PTQ model M_q with scales and zero points. The calibration dataset C.

Output: The quantization model after compensation.

```
Calculate M_q's perplexity p_o on C.
for each layer in M_q do
    Set p_n (the current best perplexity) as p_0, and k_n (the current best k value) as zero.
    Calculate discarded weights D for each FC in this transformer layer.
    Pre-design a set of search space dimensions.
    for dimension k in the search space do
        Uniformly update all FC layers in the model layer by Equation 4, or 5, or 10 using this k value.
        Calculate M_q's perplexity p with the updated FC layers in the calibration dataset C.
        if p < p_n then
            Update p_n with p and k_n with k.
        end if
    end for
    if p_n < p_o then
        Based on dimension k_n and Equation 4, or 5, or 10, update M_q's layer.
        Set p_o as p_n.
    end if
end for
```

a memory-greedy process, in practice we adaptively update $\mathbb{E}[\Delta y \Delta y^\top]$ and $\mathbb{E}[\Delta y]$ in a streaming fashion.

It is worth noting that we proposed 3 strategies to construct the search space by low-rank technology, because low-rank decomposition is easy to implement and can quickly get an approximation of D without breaking model output. There may exist other generation algorithms, too.

But, as we will show later, *what helps quantization is our idea to search in a search space, plus our DWR framework, rather than any specific search space*—even when the random search space is used, DWR can achieve higher accuracy than the original LLM quantization algorithm.

Discarded Weight Recycling

Now we propose our Discarded Weight Recycling (DWR) framework after obtaining a search space. We first perform the PTQ algorithm, and then run our DWR framework. In particular, previous LLM PTQ algorithms will first sample a small calibration set to help correct the model's output. Here we continue to use the same calibration set to run our framework and show DWR in Algorithm 1.

As the algorithm shows, our framework first calculates the original quantization model's perplexity. Then, according to the pre-designed search space, we update all FC weights in a layer of LLM and calculate the updated model's perplexity on the calibration set. In the end, we find the best-performing weights in the search space and compare the updated model and the original quantization model. If the updated model has a lower perplexity, we accept the updated FC layers. Otherwise, we use the original FC weights. We layer-by-layer recycle the discarded weights and this process is repeated until the last model layer. For simplicity, when applying the random generation strategy, we will fix the R matrix at the corresponding location in each LLM layer. For example, the QKV FC layers in different LLM attention blocks will share the same R matrix, as will in other FC layers. When using AFM generation, we will first update the previous layer, and then calculate $\text{Cov}(\Delta y)$ for each FC weights in the next layer. Note that when calculating $\text{Cov}(\Delta y)$, the input x is not the same between the compensated quantization model and the original model. Our purpose here is just to generate the weight search space, so we ignore this difference and use the original model's input x.

Therefore, our DWR framework can easily integrate with various search space generation methods, as long as they provide suitable potential updates. It indicates that our approach is not dependent on a specific search space and thus has great flexibility. Our DWR framework does not rely on any quantization algorithms. We will show that all three strategies can achieve better performances than the original quantization model. We use the quantization model's perplexity as a criterion. This strategy introduces global information and solves the problem of deciding whether to accept discarded weight, which is the most important factor in improving LLM's accuracy. If there are enough computing resources, one can further conduct QAT algorithms after DWR, which is better than directly fine-tuning PTQ model.

Experiments

We now evaluate our methods in this section. We first introduce experiment settings and then present main experimental results. We end this section with several analyses. All experiments are conducted with PyTorch.

Settings

Foundation models. We evaluate our framework on the 7B1 model of BLOOM (Le Scao et al. 2022), and the 7B, 13B and 70B models of LLaMA2 (Touvron et al. 2023), and LLaMA3-8B (Dubey et al. 2024).

Quantization. For PTQ methods, we adopt GPTQ (Frantar et al. 2022) with INT4 & 3 weight quantization in BLOOM, LLaMA2 and LLaMA3 families. Since GPTQ cannot handle weight-only INT2 quantization, we also perform OmniQuant (Shao et al. 2023) with INT2 weight-only quantization in LLaMA2 families. Our approach is also compatible with other PTQ methods such as AWQ (Lin et al. 2024) and SPQR (Dettmers et al. 2024b). We conduct a group-wise asymmetric quantization (with a group size of 128) in the PTQ experiments. In particular, GPTQ and OmniQuant take 128 samples from the C4 and WikiText2 datasets as calibration sets respectively, and each sample is 2048 tokens long. We use the same calibration set when performing DWR after these PTQ algorithms. The default search space method we apply is AFM generation. Note that the linear layer in LLaMA2 does not contain bias, so we do not update bias when applying AFM generation. In practice, we will skip compensating the first 1/6 block because the error accumulation in those blocks is so small that

Method	Bits	W2 (↓)	C4 (↓)	BoolQ	PIQA	SIQA	HLSW	WG	ARC-e	ARC-c	OBQA	Avg.
BLOOM-7B1	16	11.37	14.12	62.78	73.50	33.37	62.32	64.40	57.37	33.45	36.00	52.90
GPTQ	4	11.49	14.23	62.72	73.45	33.27	61.18	63.22	55.60	33.31	35.80	52.32
+DWR	4	**11.44**	**14.17**	63.15	73.61	33.27	62.12	64.48	57.37	33.45	36.20	**52.96**
GPTQ	3	11.97	14.70	62.72	71.60	33.06	59.96	61.25	54.59	32.42	34.20	51.23
+DWR	3	**11.72**	**14.47**	63.27	72.85	33.37	60.57	62.83	56.69	32.68	33.60	**51.98**
LLaMA2-7B	16	5.47	6.98	77.77	79.05	32.91	76.00	69.22	74.58	46.25	44.20	62.50
GPTQ	4	5.70	7.24	76.73	78.51	32.91	75.47	68.27	73.36	44.54	42.20	61.50
+DWR	4	**5.49**	**6.98**	77.28	78.94	32.91	75.85	68.82	74.66	45.90	43.80	**62.27**
GPTQ	3	6.42	7.94	72.84	76.66	33.27	71.89	68.19	68.69	40.78	40.60	59.12
+DWR	3	**5.52**	**7.01**	76.82	78.84	32.91	75.59	69.14	74.45	45.90	44.00	**62.21**
OmniQuant	2	11.23	15.45	60.15	66.65	32.19	51.93	56.51	45.54	27.22	30.60	46.35
+DWR	2	**10.41**	**15.19**	61.99	66.97	32.91	51.07	56.35	46.80	29.01	31.40	**47.06**
LLaMA2-13B	16	4.88	6.47	80.61	80.52	33.11	79.38	72.30	77.40	49.06	45.20	64.70
GPTQ	4	4.99	6.57	78.72	80.41	33.42	79.03	71.98	76.68	49.06	44.40	64.21
+DWR	4	**4.90**	**6.47**	80.61	80.47	33.11	79.31	72.38	77.48	49.40	45.20	**64.75**
GPTQ	3	5.45	7.05	77.89	79.00	32.91	76.78	70.79	74.45	44.62	42.00	62.31
+DWR	3	**4.97**	**6.53**	80.73	80.63	33.01	79.24	72.77	77.65	49.32	45.40	**64.84**
OmniQuant	2	8.33	11.15	64.89	70.67	32.70	59.15	57.85	56.99	33.70	35.00	51.37
+DWR	2	**7.78**	**11.06**	65.90	70.18	32.60	59.36	57.77	56.52	33.87	35.60	**51.48**
LLaMA2-70B	16	3.32	5.52	83.70	82.75	33.11	83.81	77.98	80.98	57.34	48.80	68.56
GPTQ	4	3.42	5.59	83.03	82.48	32.91	83.47	77.27	80.56	57.08	48.20	68.13
+DWR	4	**3.38**	**5.53**	83.46	82.64	33.04	83.56	77.81	80.84	57.33	48.60	**68.41**
GPTQ	3	3.88	5.88	82.20	81.99	33.01	82.14	76.87	79.34	55.55	48.80	67.49
+DWR	3	**3.40**	**5.58**	82.87	81.99	33.06	82.42	77.58	79.63	55.97	49.20	**67.85**
OmniQuant	2	6.54	8.53	71.16	75.30	32.96	70.80	69.46	68.18	41.38	38.00	58.40
+DWR	2	**6.27**	**8.42**	71.50	75.63	32.96	70.85	68.90	68.56	41.72	38.80	**58.62**
LLaMA3-8B	16	6.24	8.96	82.17	81.18	32.91	78.93	73.95	81.14	53.50	45.00	66.10
GPTQ	4	9.96	11.76	77.46	73.29	32.50	74.83	72.93	64.06	38.48	45.60	59.89
+DWR	4	**6.26**	**8.97**	81.83	81.34	32.91	78.86	74.35	81.36	52.99	44.20	**65.98**
GPTQ	3	67.06	44.64	64.59	71.33	33.06	63.25	66.69	66.12	40.70	39.20	55.62
+DWR	3	**6.34**	**9.05**	82.08	81.12	32.91	78.61	73.80	80.72	52.13	46.60	**66.00**

Table 1: Results in WikiText2, C4 and zero-shot commonsense QA datasets with PTQ algorithms.

compensating them does not affect the results. For the selection range of dimension k, we set the search interval to 512 on the 7B models, 1024 on the 13B and 30B models, and 2048 on the 70B model. For example, the hidden sizes of BLOOM-7B1 are 4096, and we set the candidate values of k are $\{512, 1024, \cdots, 4096\}$. A large search dimension interval helps to improve the speed of the algorithm.

For QAT methods, we follow the settings of QA-LoRA (Xu et al. 2024) and fine-tune LLaMA2-7B & 13B models with INT4 & 3 quantization. That is, based on the GPTQ INT4 & 3 quantization models, we will directly fine-tune models or first perform DWR and then fine-tune them. All training hyperparameters are the same as the original QA-LoRA paper, and we randomly sample 23k data from Flanv2 (Longpre et al. 2023) dataset to fine-tune the quantization models.

Evaluation metrics. Following the settings of GPTQ and OmniQuant, we evaluate the perplexity on the WikiText2 (Stephen et al. 2017) and C4 (Raffel et al. 2020) datasets. We further assess the zero-shot common sense question answering (QA) ability on tasks covering SIQA (Sap et al. 2019), HellaSwag (Zellers et al. 2019), PIQA (Bisk et al. 2020), WinoGrande (Sakaguchi et al. 2021), ARC (Clark et al. 2018), BoolQ (Clark et al. 2019), and OpenBookQA (Mihaylov et al. 2018). We also evaluate both the zero-shot and five-shot performance of the LLMs on massively multitask language understanding (MMLU) benchmark (Hendrycks et al. 2021). It consists of 57 language tasks including humanities, STEM, social science, etc. Note that we do not report the accuracy of BLOOM series models on MMLU datasets, because even the accuracy of baseline models is close to random guesses. Therefore, those models' results on MMLU datasets provide meaningless references. We adopt lm-eval-harness (Gao et al. 2021) to produce the accuracy results.

Main Results

PTQ Results. We first apply DWR after performing GPTQ and OmniQuant. Table 1 summarizes the results of different models, bit widths in WikiText2, C4 and eight common sense reasoning datasets. The results of MMLU datasets are shown in Table 2. Each block is based on the same foundation model specified in the first row. Note that we abbreviate WikiText2, HellaSwag, WinoGrande, and OpenBookQA to W2, HLSW, WG, and OBQA, respectively. ARC-e and ARC-c stand for ARC-easy and ARC-challenge tasks. As the results show, our DWR-optimized models will not overfit the calibration dataset and consistently outperform the original PTQ models. The advantage is even more significant when the model size is smaller (e.g., 7B and 13B) or

Method	Bits	MMLU (0-shot)					MMLU (5-shot)				
		Hums.	STEM	Social	Other	Avg.	Hums.	STEM	Social	Other	Avg.
LLaMA2-7B	16	39.64	34.25	47.35	47.18	41.79	43.32	36.98	51.77	52.69	45.82
GPTQ	4	36.77	35.49	44.95	44.38	40.40	42.85	32.23	52.14	50.18	45.25
+DWR	4	39.38	34.22	46.64	46.77	**41.45**	43.00	36.76	51.25	52.59	**45.53**
GPTQ	3	34.26	31.02	39.32	41.17	36.44	38.96	35.05	43.84	47.28	40.99
+DWR	3	39.30	34.63	47.09	46.86	**41.63**	43.00	37.65	51.19	52.20	**45.63**
OmniQuant	2	25.23	22.52	25.38	25.59	24.73	24.17	28.39	31.39	25.72	**27.04**
+DWR	2	26.31	24.45	23.40	23.95	**24.74**	24.44	28.20	30.42	26.13	26.97
LLaMA2-13B	16	47.99	42.21	61.23	59.41	52.12	53.43	43.83	63.21	61.35	55.17
GPTQ	4	46.63	42.25	59.70	58.71	51.18	50.48	43.77	62.37	61.96	54.12
+DWR	4	48.16	42.66	61.23	59.29	**52.25**	53.41	44.21	62.98	61.18	**55.16**
GPTQ	3	43.95	38.98	54.47	53.07	47.16	47.82	42.12	58.27	56.58	50.77
+DWR	3	46.55	41.83	58.82	56.32	**50.34**	51.46	43.93	60.77	59.58	**53.60**
OmniQuant	2	25.14	22.84	23.30	25.43	24.28	26.72	28.54	30.22	32.02	29.07
+DWR	2	25.72	24.39	24.18	26.68	**25.30**	26.89	29.94	29.96	32.64	**29.52**
LLaMA2-70B	16	60.70	53.66	77.67	72.45	65.44	64.53	57.66	79.46	74.93	68.56
GPTQ	4	59.57	53.19	76.37	71.26	64.41	64.21	57.47	79.16	74.67	68.24
+DWR	4	59.49	53.47	76.47	71.52	**64.52**	64.36	57.56	79.49	74.99	**68.50**
GPTQ	3	58.11	53.12	73.77	68.68	62.76	61.40	54.71	78.84	73.38	**66.37**
+DWR	3	58.81	51.70	74.16	70.49	**63.16**	61.45	55.34	78.19	73.25	66.36
OmniQuant	2	35.24	32.00	39.84	38.91	36.33	40.19	36.73	49.11	48.31	43.16
+DWR	2	35.11	33.75	42.25	41.10	**37.69**	40.87	36.54	50.18	48.95	**43.73**
LLaMA3-8B	16	57.34	55.06	74.07	70.42	63.39	59.94	55.82	76.31	72.16	65.30
GPTQ	4	55.41	53.28	72.31	68.59	61.55	58.41	54.30	74.52	70.52	63.69
+DWR	4	57.05	55.03	74.00	70.74	**63.34**	59.60	55.95	75.95	71.71	**65.04**
GPTQ	3	49.37	45.86	63.37	60.89	54.20	51.69	47.19	65.13	61.86	55.88
+DWR	3	56.24	53.76	72.73	69.55	**62.24**	60.30	55.28	76.18	71.71	**65.18**

Table 2: Accuracy in MMLU datasets with PTQ algorithms.

Method	Bits	W2 (↓)	C4 (↓)	BoolQ	PIQA	SIQA	HLSW	WG	ARC-e	ARC-c	OBQA	Avg.
LLaMA2-7B	16	5.47	6.98	77.77	79.05	32.91	76.00	69.22	74.58	46.25	44.20	62.50
QA-LoRA	4	5.54	7.12	77.74	78.99	32.80	75.63	68.76	74.16	45.22	44.20	62.19
+DWR	4	**5.52**	**7.05**	78.41	79.02	32.96	75.94	69.10	74.37	45.30	45.00	**62.51**
QA-LoRA	3	5.68	7.19	74.40	77.65	31.99	74.59	68.42	72.60	44.11	42.20	60.75
+DWR	3	**5.64**	**7.12**	74.98	77.63	31.93	74.98	68.93	72.90	44.28	42.80	**61.05**
LLaMA2-13B	16	4.88	6.47	80.61	80.52	33.11	79.38	72.30	77.40	49.06	45.20	64.70
QA-LoRA	4	4.95	6.51	79.24	80.22	33.52	78.99	72.10	76.98	49.15	45.00	64.40
+DWR	4	**4.93**	**6.49**	80.31	80.48	33.62	79.21	72.04	77.15	49.32	45.40	**69.69**
QA-LoRA	3	5.17	5.72	78.32	79.36	32.96	77.88	71.02	75.67	47.27	44.20	63.34
+DWR	3	**5.04**	**5.69**	78.75	79.56	33.06	77.91	71.48	75.59	47.53	44.80	**63.59**

Table 3: Results in WikiText2, C4 and zero-shot commonsense QA datasets with QA-LoRA.

Method	Bits	MMLU (0-shot)					MMLU (5-shot)				
		Hums.	STEM	Social	Other	Avg.	Hums.	STEM	Social	Other	Avg.
LLaMA2-7B	16	39.64	34.25	47.35	47.18	41.79	43.32	36.98	51.77	52.69	45.82
QA-LoRA	4	42.23	36.23	48.31	49.22	43.60	42.19	37.34	48.37	49.25	43.80
+DWR	4	43.77	38.56	50.16	50.08	**45.64**	43.71	39.23	50.13	50.11	**46.95**
QA-LoRA	3	38.71	34.92	45.68	45.76	40.66	38.78	35.85	46.26	45.76	40.97
+DWR	3	40.23	37.02	47.76	48.23	**42.57**	40.19	37.83	47.85	48.28	**42.73**
LLaMA2-13B	16	47.99	42.21	61.23	59.41	52.12	53.43	43.84	63.21	61.35	55.17
QA-LoRA	4	47.92	43.43	61.55	59.36	51.82	51.02	43.83	62.71	60.83	53.75
+DWR	4	48.35	43.75	62.07	59.42	**52.17**	51.64	43.78	62.97	61.14	**54.12**
QA-LoRA	3	46.89	41.97	59.21	58.25	50.45	49.87	42.32	60.45	59.72	52.34
+DWR	3	47.29	42.29	60.05	58.31	**50.85**	50.41	42.55	61.58	59.97	**52.87**

Table 4: Accuracy in MMLU datasets with QA-LoRA.

the bit width is lower (e.g., INT3 or even INT2). This phenomenon indicates that DWR is a powerful solution in scenarios where computational efficiency is required. In some cases, the compensated model's INT4 accuracy is even bet-

Method	INT4			INT3		
	W2	C4	ACC.	W2	C4	ACC.
GPTQ	11.49	14.23	52.32	11.97	14.70	51.23
Random	11.47	14.21	52.49	11.75	11.46	51.71
SVD	**11.42**	**14.16**	52.65	**11.45**	**14.21**	**52.60**
AFM	11.44	14.17	**52.96**	11.72	14.47	51.98

Table 5: Results in WikiText2, C4 and common sense QA datasets with different search space generation methods.

Method	INT4			INT3		
	W2	C4	ACC.	W2	C4	ACC.
GPTQ	11.49	14.23	52.32	11.97	14.70	51.23
RTN	11.58	14.31	52.35	12.53	15.21	51.23
KL	11.55	14.31	52.41	12.25	15.10	51.54
PPL	**11.44**	**14.17**	**52.96**	**11.72**	**14.47**	**51.98**

Table 6: Results in WikiText2, C4 and common sense QA datasets with different evaluation criteria.

ter than the 16-bit original model. In particular, we find that GPTQ could not deal with LLaMA3 well, and there was a large accuracy loss with INT4/3 quantization. In contrast, our method can obtain a quantized model with higher accuracy on the basis of GPTQ. These results reveal the effectiveness of our algorithm.

QAT Results. We apply QA-LoRA to fine-tune GPTQ and our compensated INT4 & 3 quantized LLaMA2-7B & 13B models, respectively. The results of WikiText2, C4 and eight common sense question answering datasets are shown in Table 3. The accuracies of MMLU datasets are shown in Table 4. We can obtain similar conclusions from these experiments. Those results indicate that our DWR does not rely on any exact quantization algorithms, and can change the traditional quantization paradigm, i.e., first PTQ and then QAT. DWR can be used as a plug-and-play framework and is important in boosting low-bit quantization LLMs.

Ablation Studies

We perform several analyses in this section to explore the impact of different modules of our method.

Effect of search spaces. Our default search space method is AFM generation. Here we study the effect of different search spaces. We apply INT4 & 3 BLOOM-7B1 with GPTQ quantization as baselines, and other settings remain consistent. The results in Table 5 show that even the randomly generated search space can achieve better results than the original quantized model, which suggests that our whole proposed framework is the reason for its effectiveness, rather than a specific search space generation algorithm.

Effect of evaluation criterion. DWR applies LLM's perplexity as the evaluation criterion. Now we compare it with other strategies, i.e., direct merging without any criterion (namely RTN quantization), and the KL value between the output of the quantization model and the original model. We use BLOOM-7B1 model with INT4 & 3 quantization. The results are shown in Table 6, and directly using perplexity achieves the highest precision.

Effect of hyperparameters. In the original settings of BLOOM-7B1 with INT4 quantization, we calculate perplexity by 128 samples, skip the first 1/6 blocks, and set the searching interval to 512. Here we study the influence of these hyperparameters. The results shown in Table 7 indicate that with various hyperparameters, DWR can achieve higher precision than the original quantization model. Therefore, DWR framework is robust for hyperparameters.

#Data	#Block	#Interval	W2	C4	ACC.
Baseline			11.49	14.23	52.32
8	5	512	11.45	14.20	52.67
128	5	512	11.44	14.17	**52.96**
256	5	512	**11.40**	**14.15**	52.69
128	0	512	11.42	14.18	52.60
128	10	512	11.43	14.17	52.69
128	5	256	11.42	14.16	52.78
128	5	1024	11.42	14.16	52.77

Table 7: Results in WikiText2, C4 and common sense QA datasets with different hyperparameters.

Model	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B
Time	1.12	2.67	6.21

Table 8: The running time (hours) of our DWR algorithm with LLaMA2 families.

The running time of DWR. Here we report the time required to run our DWR algorithm. We take the INT4 quantized LLaMA2 models as examples. In particular, we run LLaMA2-7B & 13B on 8 Tesla V100 GPUs and LLaMA2-70B on 4 80G A100 GPUs. The results are shown in Table 8. It can be seen that the runtime of our DWR algorithm is acceptable. It is worth noting that our DWR can perform search operations in parallel on multiple GPUs at the same time, and then we uniformly judge the optimal solution and update one layer after the search process. Therefore, we can use multiple GPUs simultaneously to achieve fast speed and avoid abundant communication overhead, which is not done by the previous LLM PTQ and QAT algorithms.

Conclusion, Limitation, and Future Work

In this paper, we proposed a plug-and-play framework to recycle the treasures in discarded weights caused by LLM quantization. With various search spaces, we designed a discarded weights recycling framework, named DWR. Our DWR consistently achieves notable and stable improvements on various LLM evaluation datasets. Besides, DWR is insensitive to hyperparameters, achieves notable and stable improvements on various LLM evaluation datasets, and can be easily combined with various LLM PTQ and QAT algorithms without bringing additional inference burden.

Although DWR can significantly improve quantization LLM's precision, we currently only consider weight-only quantization scenarios. Therefore, an interesting direction is how to extend DWR to activation quantization, which we leave for future work. In addition, we will also extend our algorithm to more models and tasks, such as multimodal large models and LLMs with MoE architecture. We propose these as areas for future exploration.

Acknowledgments

This work was supported by Alibaba Research Intern Program and the National Natural Science Foundation of China under Grant 62276123. We sincerely thank Zipeng Zhang at Alibaba Inc. and Jie Shao at NJU for their selfless help during the experiment.

References

Bisk, Y.; Zellers, R.; Bras, R. L.; Gao, J.; and Choi, Y. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 7432–7439.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, volume 33, 1877–1901.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2924–2936.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Dettmers, T.; Lewis, M.; Belkada, Y.; and Zettlemoyer, L. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems 35*, volume 35, 30318–30332.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2024a. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems 36*, volume 36, 10088–10115.

Dettmers, T.; Svirschevski, R. A.; Egiazarian, V.; Kuznedelev, D.; Frantar, E.; Ashkboos, S.; Borzunov, A.; Hoefler, T.; and Alistarh, D. 2024b. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The LLaMA 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. GPTQ: Accurate post-training compression for generative pretrained transformers. In *International Conference on Learning Representations*.

Gao, L.; Tow, J.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; McDonell, K.; Muennighoff, N.; et al. 2021. A framework for few-shot language model evaluation.

Golub, G. H.; and Van Loan, C. F. 2013. *Matrix computations*. Johns Hopkins University Press.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, 1737–1746.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2021. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Kim, J.; Lee, J. H.; Kim, S.; Park, J.; Yoo, K. M.; Kwon, S. J.; and Lee, D. 2024. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In *Advances in Neural Information Processing Systems 36*, volume 36, 36187–36207.

Le Scao, T.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A. S.; et al. 2022. BLOOM: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 87–100.

Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; and Chandra, V. 2024. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics*, 467–484.

Longpre, S.; Hou, L.; Vu, T.; Webson, A.; Chung, H. W.; Tay, Y.; Zhou, D.; Le, Q. V.; Zoph, B.; Wei, J.; et al. 2023. The Flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, 22631–22648.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2381–2391.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Sakaguchi, K.; Bras, R. L.; Bhagavatula, C.; and Choi, Y. 2021. WinoGrande: An adversarial winograd schema challenge at scale. In *Communications of the ACM*, 99–106.

Sap, M.; Rashkin, H.; Chen, D.; Le Bras, R.; and Choi, Y. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, 4463–4473.

Shao, W.; Chen, M.; Zhang, Z.; Xu, P.; Zhao, L.; Li, Z.; Zhang, K.; Gao, P.; Qiao, Y.; and Luo, P. 2023. OmniQuant:

Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*.

Stephen, M.; Caiming, X.; James, B.; Richard, S.; et al. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; et al. 2023. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, volume 30, 5998–6008.

Wu, J. 2020. *Essentials of pattern recognition: An accessible approach*. Cambridge University Press.

Xiao, G.; Lin, J.; Seznec, M.; Wu, H.; Demouth, J.; and Han, S. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 38087–38099. PMLR.

Xu, Y.; Xie, L.; Gu, X.; Chen, X.; Chang, H.; Zhang, H.; Chen, Z.; Zhang, X.; and Tian, Q. 2024. QA-LoRA: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yao, Z.; Wu, X.; Li, C.; Youn, S.; and He, Y. 2024. Exploring post-training quantization in LLMs from comprehensive study to low rank compensation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 19377–19385.

Yu, H.; and Wu, J. 2023. Compressing transformers: Features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11007–11015.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4791–4800.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; et al. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.