DiscQuant: A Quantization Method for Neural Networks Inspired by Discrepancy Theory

Jerry Chee JERRYCHEE@CS.CORNELL.EDU
Department of Computer Science, Cornell University

Arturs Backurs ARTURS.BACKURS@MICROSOFT.COM
Microsoft Research

Rainie Heck RAINER.HECK@TTK.ELTE.HU
Department of Mathematics, Eötvös Loránd University

Li Zhang LIZHANG4@MICROSOFT.COM
Microsoft

Janardhan Kulkarni JAKUL@MICROSOFT.COM
Microsoft Research

Thomas Rothvoss ROTHVOSS@UW.EDU
Department of Mathematics, University of Washington

Sivakanth Gopi SIGOPI@MICROSOFT.COM
Microsoft Research

Editors: Nika Haghtalab and Ankur Moitra

Abstract

Quantizing the weights of a neural network has two steps: (1) Finding a good low bit-complexity representation for weights (which we call the quantization grid) and (2) Rounding the original weights to values in the quantization grid. In this paper, we study the problem of rounding optimally given any quantization grid. The simplest and most commonly used way to round is Round-to-Nearest (RTN). By rounding in a data-dependent way instead, one can improve the quality of the quantized model significantly.

We study the rounding problem from the lens of *discrepancy theory*, which studies how well we can round a continuous solution to a discrete solution without affecting solution quality too much. We prove that given $m = \mathrm{poly}\left(\frac{\log n}{\varepsilon}\right)$ samples from the data distribution, we can round nearly all n model parameters such that the expected approximation error of the quantized model on the true data distribution is $\leq \varepsilon$ as long as the space of gradients of the original model is approximately low rank (which we empirically validate). Our algorithm is based on the famous Lovett-Meka algorithm from discrepancy theory and uses sticky Brownian motion to find a good rounding.

We also give a simple and practical rounding algorithm called *DiscQuant*, which is inspired by our theoretical insights. In our experiments, we demonstrate that DiscQuant significantly improves over the prior state-of-the-art rounding method called GPTQ and the baseline RTN over a range of benchmarks on Phi3mini-3.8B and Llama3.1-8B. For example, rounding Phi3mini-3.8B to a fixed quantization grid with 3.25 bits per parameter using DiscQuant gets 64% accuracy on the GSM8k dataset, whereas GPTQ achieves 54% and RTN achieves 31% (the original model achieves 84%). We make our code available at https://github.com/jerry-chee/DiscQuant.

Keywords: Quantization, Discrepancy Theory, Rounding, LLM, Neural Networks

1. Introduction

Modern deep learning models continue to grow in size, incurring greater challenges to train and serve these models. Post training compression methods have emerged which aim to make model inference faster and cheaper. Compressing after pretraining is desirable among practitioners who either cannot afford to train models themselves, or do not want to change the expensive training process too much. In this paper, we study post training quantization (PTQ) of the model weights. Quantization reduces the memory requirements of the model, and speeds up inference for LLMs under memory-bound settings such as the generation phase (as opposed to prefilling phase which is compute-bound) (Kwon et al., 2023).

The quantization problem can be divided into two overall steps: (1) Construct a good low bit-complexity representation for the weights (we colloquially call this the quantization grid), and (2) Round the original weights to values in the quantization grid. Within step (1), we also consider those methods which apply a transformation on the weights to better match the encoding format. There has been much recent work on weights-only PTQ for LLMs. To date, the vast majority of such research has been focused on step (1): constructing good low bit representations (Shao et al., 2024; Tseng et al., 2024a; Egiazarian et al., 2024). However, work on rounding methods is under-explored. To the best of our knowledge, Round-to-Nearest (RTN) and GPTQ (Hassibi et al., 1993; Frantar et al., 2022, 2023) are the primary rounding methods for LLM weight quantization. RTN is a simple baseline, and GPTQ is a data dependent method which aims to match the activations of the quantized model with that of the original model layer-by-layer. We now introduce our method.

Let $f(w; s)$ be the loss function of a pre-trained neural network with weights $w \in \mathbb{R}^n$ on an input sample s and let $\mathcal{D}_{\text{data}}$ be the sample data distribution. Suppose we are also given a (scalar) quantization grid $\mathcal{Q} = Q_1 \times Q_2 \times \cdots \times Q_n$ where $Q_j \subset \mathbb{R}$ is a finite set of quantization points available to quantize the j^{th} parameter.[1] In this work, we focus on scalar quantization which allows us to write the quantization grid as a product set, i.e., each parameter can be independently rounded to a finite set of available values. Alternatively, in vector quantization a group of d variables are rounded together to one of a finite set of quantization points in $\mathbb{R}^d$, which has been used in some prior works (Tseng et al., 2024a; Egiazarian et al., 2024; van Baalen et al., 2024). Generalizing our method to vector quantizers is an interesting future research direction.

Our goal is to find a rounding $\hat{w} \in \mathcal{Q}$ of the original weights w such $f(\hat{w}; s) \approx f(w; s)$ where $s \sim \mathcal{D}_{\text{data}}$. We further impose the constraint that for each parameter w_j, we only round down or round up to the available values in Q_j. Therefore we only have two choices for $\hat{w}_j$ denoted by $w_j^{\text{down}}, w_j^{\text{up}} \in Q_j$ where $w_j^{\text{down}} \leq w_j \leq w_j^{\text{up}}$ are the closest points below and above w_j in Q_j respectively.[2] We can picture the set of allowed $\hat{w}$ as vertices of the hypercube $H = \prod_{j=1}^{n} [w_j^{\text{down}}, w_j^{\text{up}}]$ as shown in Figure 1. We make this assumption because we don't want to change any parameter of the original model too much during quantization, and we consider it an important property of algorithms we design. We can approximate the quantization error Δf using Taylor expansion as:

$$\Delta f = f(\hat{w}; s) - f(w; s) \approx \langle \nabla_w f(w; s), \hat{w} - w \rangle \tag{1}$$

Here we are assuming that the quantization grid $\mathcal{Q}$ is fine enough and since we only round each parameter up or down, $\|\hat{w} - w\|$ is small enough to ignore the higher order terms. Figure 2 demon-

1. The quantization grid $\mathcal{Q}$ can depend on w, like in Block Scaling (Frantar et al., 2023). So ideally, we should write $\mathcal{Q}_w$, but we ignore the dependence to simplify notation.
2. If $w_j < \min Q_j$ or $w_j > \max Q_j$, we just set $w_j^{\text{up}} = w_j^{\text{down}} = \min Q_j$ or $\max Q_j$ respectively.

strates the empirical validity of this assumption by showing that the error term Δf is well-correlated with the first order approximation $\langle \nabla_w f(w;s), \hat{w} - w \rangle$.

Some prior works such as Nagel et al. (2020); Hassibi et al. (1993); LeCun et al. (1989) have assumed that the first order term can be assumed to be zero because the model is trained to convergence and therefore focused on reducing the second order term. But the model being trained to convergence only implies that the mean of the gradient distribution over many input samples is nearly zero. The gradient distribution still has large variance and individual gradients are not small. Table 1 shows the squared mean $\|\mathbb{E}(g)\|^2$ and variance $\mathbb{E}\|g\|^2$ of per-sample gradients for Phi3-mini-128k and Llama3.1-8B models. As expected the mean gradients are small due to pretraining convergence, but the variance is much larger. Therefore to reduce the quantization error Δf, it is enough to reduce the first order term. The following is our main result stated informally, see Theorem 2 for the formal statement.

Model	$\|\mathbb{E}(g)\|^2$	$\mathbb{E}\|g\|^2$
Phi3-mini-128k	0.1021	4.7812
Llama3.1-8B	1.6328	107

Table 1: $\|\mathbb{E}(g)\|^2$ vs $\mathbb{E}\|g\|^2$ over 8192 samples from RedPajama-1T-Sample dataset with window size 2048.

Theorem 1 (Informal) *If the eigenvalues of the covariance matrix of gradients decay polynomially fast, then given* $m = \text{poly}\left(\frac{\log n}{\varepsilon}\right)$ *samples* $s_1, s_2, \ldots, s_m \sim \mathcal{D}_{data}$ *there is an polynomial time randomized algorithm to find* $\hat{w}$ *with* $n - m$ *weights rounded such that* $\mathbb{E}_{s \sim \mathcal{D}_{data}}[| \langle \nabla_w f(w;s), \hat{w} - w \rangle |] \leq \varepsilon$.

Note that as per Chinchilla optimal scaling laws (Hoffmann et al., 2022), the number of pretraining tokens scale roughly proportional to the number of model parameters n. Therefore the number of pretraining steps scales with n, but our theorem shows that we can achieve small quantization error via PTQ in only $\text{poly}(\log n)$ steps which is negligible compared to pretraining time. Note, having more samples than required in Theorem 1 doesn't hurt or help reduce the quantization error further. Our theoretical result gives a mathematical reason for why a good rounding should exist. Even proving the existence of a good rounding is non-trivial; we don't know of a simpler proof of this (i.e., without using discrepancy theory) even if we don't care about a polynomial time algorithm.

1.1. Proof Sketch

Define the squared quantization error as

$$\text{QE}(\hat{w}) = \mathbb{E}_{s \sim \mathcal{D}_{\text{data}}}[\langle \nabla_w f(w;s), \hat{w} - w \rangle^2].$$

Our goal is to find a rounding $\hat{w}$ such that $\text{QE}(\hat{w}) \approx 0$. Suppose we are given m samples $s_1, s_2, \ldots, s_m \sim \mathcal{D}_{\text{data}}$ sampled independently from the data distribution, where $m \ll n$. Define the empirical squared quantization error as

$$\widehat{\text{QE}}(\hat{w}) = \mathbb{E}_{i \in [m]}[\langle \nabla_w f(w;s_i), \hat{w} - w \rangle^2].$$

We now break our task of bounding $\text{QE}(\hat{w})$ into two parts of bounding the empirical error and generalization error as follows:

$$\text{QE}(\hat{w}) = \underbrace{\widehat{\text{QE}}(\hat{w})}_{\text{empirical error}} + \underbrace{\text{QE}(\hat{w}) - \widehat{\text{QE}}(\hat{w})}_{\text{generalization error}}. \tag{2}$$

For simplicity, we will assume that the quantization grid is uniform and $w_i^{\text{up}} - w_i^{\text{down}} = \delta$ for all $i \in [n]$ where $\delta > 0$ is the distance between grid points. See Appendix C for how to generalize our results to non-uniform grids[3]. We will introduce new parameters $x \in [0,1]^n$ and define $w^x = w^{\text{down}} + \delta x$. Note that w_i^x interpolates between w_i^{down} and w_i^{up} where $w_i = w_i^{\text{down}}$ if $x_i = 0$ and $w_i = w_i^{\text{up}}$ if $x_i = 1$. Let $y \in [0,1]^n$ be the interpolation point corresponding to the original weights, i.e., $w^y = w$. We can rewrite the error in terms of x as follows:

$$\langle \nabla_w f(w; s_i), w^x - w \rangle = \langle \nabla_w f(w; s_i), w^x - w^y \rangle = \delta \langle \nabla_w f(w; s_i), x - y \rangle .$$

1.1.1. BOUNDING EMPIRICAL ERROR

To make $\widehat{\text{QE}}(w^x) = 0$ exactly, we will impose m linear constraints $\langle \nabla_w f(w; s_i), x - y \rangle = 0$ on x for $i \in [m]$. Let M be an $m \times n$ matrix whose i^{th} row is given by $\nabla_w f(w; s_i)$. Then the linear constraints can be simply written as $M(x - y) = 0$. Our goal is to find a fully integral $\hat{x} \in \{0,1\}^n$ such that $M(\hat{x} - y) \approx 0$. We will instead find an almost integral $\hat{x}$ such that $M(\hat{x} - y) = 0$ exactly. Let $V = \{x \in \mathbb{R}^n : Mx = My\}$ which is an affine subspace of dimension $\geq n - m$. Define $K = [0,1]^n \cap V$ as the intersection of the hypercube with this subspace. K is a convex polytope and it is non-empty because $y \in K$. Therefore any vertex of K should have $n - m$ integral coordinates (i.e., coordinates j such that $x_j \in \{0,1\}$).[4] See Figure 1 for geometric intuition about why this is true. Since $n \gg m$, any vertex of K is almost fully integral and satisfies all the m linear constraints.

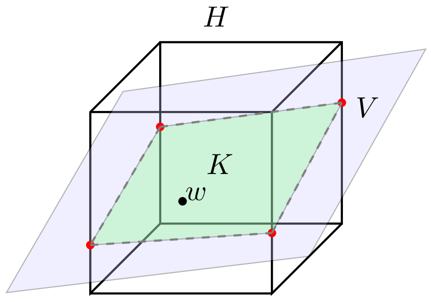

Figure 1: An illustrative figure showing the convex polytope K formed by the intersection of an n-dimensional hypercube H and an $n - m$ dimensional affine subspace V. Any vertex of K should have $n - m$ coordinates which are fully rounded.

Suppose we want a fully integral $\hat{x}$ which approximately satisfies all the m linear constraints, this precise question is answered by *discrepancy theory* which studies how to do this and relates the approximation error to properties of M such as *hereditary discrepancy* (Lovász et al., 1986; Bansal, 2022). We don't explore this direction further because the almost integral $\hat{x}$ (i.e., a vertex of K) is good enough if we apply RTN to the few remaining fractional parameters; we observe that the linear constraints are all approximately satisfied.

1.1.2. BOUNDING GENERALIZATION ERROR

To bound the generalization error, we need to show that if $\hat{x} - y$ is approximately orthogonal to m sample gradients $\nabla_w f(w; s_i)$ for $i = 1$ to m, then $\hat{x} - y$ is also orthogonal to unseen gradients $\nabla_w f(w; s)$ for samples $s \sim \mathcal{D}_{\text{data}}$. This should happen only if the gradients are approximately low rank. More precisely, let

$$\Sigma = \mathbb{E}_{s \sim \mathcal{D}_{\text{data}}}[gg^T] \text{ where } g = \nabla_w f(w; s)$$

3. In short, we will define $w^x = w^{\text{down}} \odot (1 - x) + w^{\text{up}} \odot x$ where $\odot$ is component-wise product. Note that $w_i^{\text{up}} - w_i^{\text{down}}$ is not assumed constant for all $i \in [n]$. See Appendix C for full details.
4. This is because at a vertex, we need to have n tight constraints, and V imposes only m tight constraints. So the remaining $n - m$ tight constraints should come from the hypercube. These are also called basic feasible solutions in linear programming.

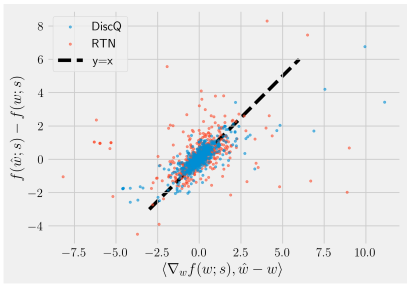

Figure 2: First order approximation vs Δf when quantizing the model to 4.25 bits using RTN and DiscQuant. Here f is the per-token loss function and s is sampled from the WikiText-2 dataset.

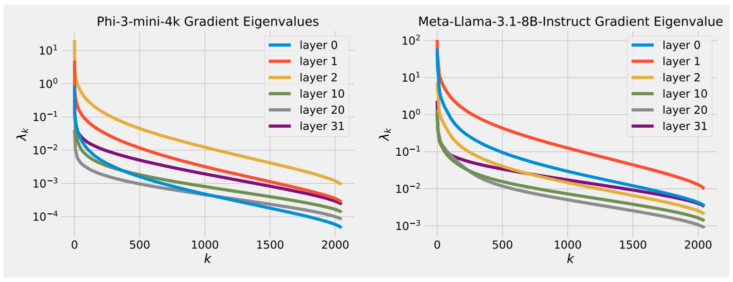

Figure 3: Eigenvalues of the covariance matrix of the gradients of pre-trained models. The covariance matrix is estimated by averaging over $8k$ sample gradients from RedPajama-1T-Sample and projecting them to 2048 dimensions using Johnson-Lindenstrauss projections.

be the covariance matrix of the distribution of sample gradients and let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be its eigenvalues. We observe that the eigenvalues decay very fast, see Figure 3 for empirical validation of this on some real world models. We model this by assuming that $\lambda_k \leq C/k^\alpha$ for $\alpha > 1$ and some constant $C > 0$. The assumption that $\alpha > 1$ is valid since

$$\mathbb{E}_s[\|g\|^2] = \mathbb{E}_s[\mathrm{Tr}(gg^T)] = \mathrm{Tr}(\mathbb{E}_s[gg^T]) = \mathrm{Tr}(\Sigma) = \sum_{i=1}^{n} \lambda_i.$$

It is well-known that the gradients of a pretrained model have bounded norm on most samples (see Table 1 for empirical validation). Therefore $\sum_{i=1}^{n} \lambda_i = \mathbb{E}_s[\|g\|^2] = O(1)$ and so the the decay coefficient $\alpha > 1$ for the series $\sum_k \frac{1}{k^\alpha}$ to converge.

Under this assumption, it is reasonable to expect generalization, but unclear how to find a generalizing solution. In fact, any deterministic algorithm which chooses $\hat{x}$ to be one of the vertices of K will most likely not generalize. We give a randomized rounding algorithm (see Algorithm 2.2) based on the famous Lovett-Meka algorithm from discrepancy theory (Lovett and Meka, 2012) which finds a vertex of K with low generalization error. The algorithm starts at y and does a random walk (Brownian motion) inside the $n - m$ dimensional subspace V formed by the linear constraints imposed by the m samples. Whenever it hits a face $x_i = 0$ or $x_i = 1$ of the hypercube, it fixes that variable and continues the random walk until almost all the variables are rounded.

1.2. DiscQuant: A Practical Algorithm

From these insights we develop a practical rounding algorithm called *DiscQuant*. The Lovett-Meka algorithm does a random walk starting from the original weights until it converges to a vertex of K. Even though Lovett-Meka is a polynomial time algorithm, it is impractical for the neural network setting where the number of parameters n is in the order of billions. Instead, we can find a vertex of K by minimizing a linear function over the convex polytope K. DiscQuant uses stochastic gradient descent to minimize two objectives, one corresponding to low Δf, and the other corresponding to minimizing a linear function. We take a knowledge distillation approach for the first term, minimizing the KL divergence between the original and quantized model. These two

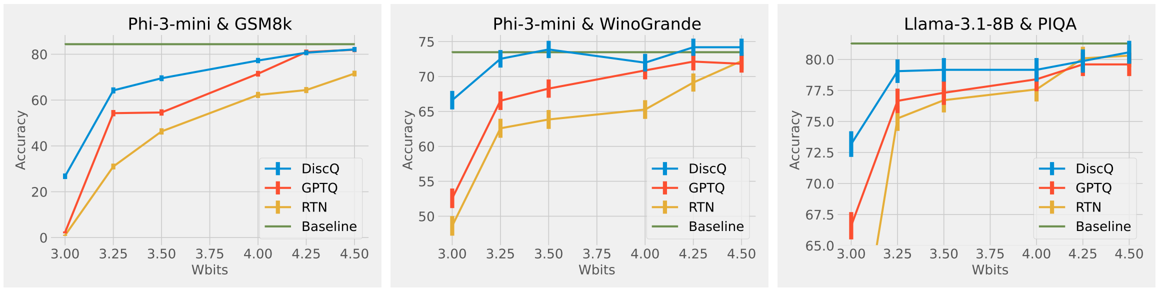

Figure 4: Select results quantizing Phi-3-mini-4k-instruct and Meta-Llama-3.1-8B-Instruct using block scaling quantization. GSM8k is a math-based generative task, and WinoGrande and PIQA are multiple choice commonsense reasoning tasks. Error bars are standard errors from lm-evaluation-harness. See Section 4 for full results.

losses are balanced with a regularization parameter $\lambda > 0$:

$$\begin{aligned} &\min_{\hat{w}} \lambda \langle c, \hat{w} \rangle + \mathbb{E}_{z \sim \mathcal{D}_{\text{data}}} \mathbb{E}_i [D_{KL} \left(p_w(\cdot | z_{<i}) \parallel p_{\hat{w}}(\cdot | z_{<i}) \right)] \\ &s.t.\ \hat{w} \in H. \end{aligned} \tag{3}$$

Here $p_w(\cdot|z_{<i})$ is the next token distribution given prefix $z_{<i}$. An astute reader may notice that the first order approximation of the KL divergence in (3) is exactly zero, and how our discussion above applies. In Section 3 where we describe in detail our exact optimization objective, we also show that the second order term approximation to KL divergence can be written as

$$D_{KL} \left(p_w(\cdot|z_{<i}) \parallel p_{\hat{w}}(\cdot|z_{<i}) \right) \approx \mathbb{E}_{t \sim p_w(\cdot|z_{<i})} \left[\langle \nabla_w \log p_w(t|z_{<i}), \hat{w} - w \rangle^2 \right].$$

So minimizing KL divergence is a succinct way to impose $\langle \nabla_w \log p_w(t|z_{<i}), \hat{w} - w \rangle \approx 0$ or equivalently $\log p_w(t|z_{<i}) \approx \log p_{\hat{w}}(t|z_{<i})$ where $t \sim p_w(\cdot|z_{<i})$ and $z \sim \mathcal{D}_{\text{data}}$. Therefore our framework still applies.

After every step of gradient descent, we project the weights back to the hypercube H. This ensures that the trajectory of DiscQuant remains close to the convex polytope K and eventually converges to a vertex of K with almost all the coordinates rounded. Instead of picking a random direction c to find a random vertex of K, we use a special c^* which let's us find the vertex closest to the original weights w (see Section 3). We use RTN to round the few unrounded parameters left at the end of the optimization.

We perform extensive experiments which show the strength of our method: on models Phi-3-mini-4k-instruct and Meta-Llama-3.1-8B-Instruct, across a variety of evaluation tasks, and across the block scaling and incoherence processing quantization formats. DiscQuant is agnostic towards the quantization grid, and can therefore be composed with other quantization methods. A subset of results can be found in Figure 4. Across tasks, models, and quantization levels, our method DiscQuant achieves superior compression over baselines GPTQ and RTN.

1.3. Related Work

Here we give a brief discussion of related work. For a more extensive discussion, see Appendix A. In this paper we focus on weights-only quantization. Quantization can also be applied to the activations or KV-cache (Ashkboos et al., 2024; Liu et al., 2024a,b). Other compression method such as pruning (Frantar and Alistarh, 2023; Sun et al., 2023) are also outside the scope of this work. To the best of our knowledge, GPTQ (Frantar et al., 2023) is the main rounding method for LLMs. It is based on the Optimal Brain Surgeon (Hassibi et al., 1993), which was adapted for pruning and quantization in Frantar et al. (2022) and then refined for quantization in GPTQ. GPTQ works by minimizing a layer-wise objective $\|WX - \hat{W}X\|_2^2$, where W is the weight matrix of a linear layer and X is the matrix of input activations to that layer (stacked as columns). Chee et al. (2023) developed a rounding algorithm which they showed was equivalent to GPTQ. Two other LLM rounding methods both use coordinate descent: Nair and Suggala (2024) has results on the open Gemma and closed PaLM-2 but with no released code, and Behdin et al. (2023) has results on the OPT, BLOOM, and Falcon model families.

There was more work on rounding methods several years ago, before the LLM boom. These papers were typically on smaller vision models. The line of work was started by AdaRound (Nagel et al., 2020) and continuing to AdaQuant (Hubara et al., 2021) and BRECQ (Li et al., 2021) employ a similar approach to ours, optimizing essentially interpolation variables between the w^{up} and w^{down} quantization grid points, while adding a concave regularization term to encourage rounding and using a rectified sigmoid to interpolate between w^{up} and w^{down}. They also do rounding layer by layer. However our method uses a linear term as a regularizer inspired from our theoretical insights using discrepancy theory and uses simple linear interpolation between w^{up} and w^{down} and we round the entire model at once.

Discrepancy theory is a deep branch of mathematics and theoretical computer science, and we refer the readers to standard textbooks for more details (Matousek, 2009; Chazelle et al., 2004; Bansal, 2022) To our knowledge, only Lybrand and Saab (2021) makes the connection between discrepancy theory and quantization. However, besides the high level motivational similarities, their work is not comparable. Lybrand and Saab (2021) reduce the problem of understanding the error introduced by quantization on the output of a *single neuron* to a problem in discrepancy. Their theoretical analysis on the generalization error only applies to quantizing the first layer of a neural network. On the other hand, we use discrepancy theory to understand when the whole network $f(w; s)$ can be approximated by $f(\hat{w}; s)$ with $\hat{w}$ in the quantization grid, and our theory holds for any network as a whole as long as our assumptions are true. Before our work, there were no theoretical bounds on the quantization error known.

2. Rounding Weights via Discrepancy Theory

In this section, we will formally state and prove our main theoretical result. In order to prove rigorous bounds, in addition to the assumption about the decay of eigenvalues of covariance matrix, we will also need a mild assumption that the distribution of gradients is well-behaved. We use the notion by O'Donnell (2014) and say that for a parameter $\beta \geq 1$, a random vector $X \in \mathbb{R}^n$ is *β-reasonable* if

$$\mathbb{E}[\langle X, \theta \rangle^4] \leq \beta \cdot \mathbb{E}[\langle X, \theta \rangle^2]^2 \quad \forall \theta \in \mathbb{R}^n.$$

For example $X \sim \{-1, 1\}^n$ and a Gaussian $X \sim N(\mathbf{0}, \Sigma)$ are both $O(1)$-reasonable. Our main theoretical result is then:

Theorem 2 *Let $\alpha > 1$ and $\beta \geq 1$ be constants and let $1 \leq m \leq \frac{n}{16}$. Let $\mathcal{D}_{grad}$ be the gradient distribution*[5] *which is a β-reasonable distribution with unknown covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$ whose Eigenvalues satisfy $\lambda_k \leq \frac{C}{k^\alpha}$ for all $k = 1, \dots, n$. Then there is a randomized polynomial time algorithm that given a $y \in [0,1]^n$ and m independent samples $g_1, \dots, g_m \sim \mathcal{D}_{grad}$, produces an $\hat{x} \in [0,1]^n$ with high probability such that*

(i) the number of fractional coordinates in $\hat{x}$ satisfies $|frac(\hat{x})| \leq 16m$ and

(ii) $\mathbb{E}_{g\sim\mathcal{D}_{grad}}[\langle g, \hat{x} - y\rangle^2] \lesssim_{\alpha,\beta} C\log(\frac{n}{m}) \cdot F_\alpha(m,n)$ where

$$F_\alpha(m,n) := \begin{cases} m^{1-\alpha} & \text{if } 1 < \alpha < \frac{3}{2} \\ \frac{\log(n)}{\sqrt{m}} & \text{if } \alpha = \frac{3}{2} \\ \frac{1}{\sqrt{m}} & \text{if } \alpha > 3/2. \end{cases}$$

Ignoring polylogarithmic factors, this means that we can find an x with all but $O(m)$ coordinates rounded and with quantization error at most $m^{-\min\{\alpha-1,1/2\}}$. The proof crucially uses the Lovett Meka algorithm, so we first describe this algorithm before we prove our result. We obtain Theorem 1 by setting $w = w^y$ and $\hat{w} = w^{\hat{x}}$ as discussed in the beginning of Section 1.1. Therefore the final quantization error in Theorem 1 is equal to the quantization error in Theorem 2 multiplied by a factor of δ which is the distance between adjacent grid points.

2.1. The Lovett Meka algorithm

A seminal result by Lovett and Meka (2012) works as follows: we are given a point $y \in [0,1]^n$ in the hypercube, vectors $v_1, \dots, v_m \in \mathbb{R}^n$ with $\|v_j\|_2 = 1$ and parameters $c_j \geq 0$ so that $\sum_{j=1}^m e^{-c_j^2/16} \leq \frac{n}{16}$. Then in randomized polynomial time one can find a point $x \in [0,1]^n$ so that $|\langle v_j, x-y\rangle| \leq c_j$ for all j and at least half the coordinates of x are integral. Their algorithm is simple and elegant: we construct x as the outcome of a random walk starting at y. Then iteratively, for some small step size $\delta > 0$ we add the outcome of a random Gaussian times δ to the current point. After hitting some constraint $x_i = 0$, $x_i = 1$ or $|\langle v_j, x-y\rangle| = c_j$, the Gaussian updates will be taken orthogonal to those normal vectors. In other words, the random walk will continue in the face of the described polytope. Still Lovett and Meka (2012) prove that performing $O(\frac{1}{\delta^2})$ updates the walk will cover enough distance so that on average $\Theta(n)$ box constraints must become tight. In our setting we only need to use parameters $c_j = 0$. However we use some properties of the Lovett-Meka algorithm that are not explicitly stated elsewhere.

Theorem 3 (Derived from Lovett and Meka (2012)) *Let $g_1, \dots, g_m \in \mathbb{R}^n$ be any vectors with $m \leq \frac{n}{16}$ and let $y \in [0,1]^n$. Then there is a randomized polynomial time algorithm $\mathcal{A}_{LM}(g_1, \dots, g_m, y)$ which outputs a sample x such that*[6]

(i) $\langle g_i, x - y\rangle = 0$ for all $i \in [m]$,

(ii) $x \in [0,1]^n$ and with probability at least $\frac{1}{10}$ one has $|\{j \in [n] : x_j \in \{0,1\}\}| \geq \frac{n}{2}$ and

5. $\mathcal{D}_{\text{grad}}$ is the distribution of gradients $\nabla_w f(w; s)$ where $s \sim \mathcal{D}_{\text{data}}$.
6. Here we denote $\|M\|_{\mathcal{S}(1)}$ as the sum of the singular values of a matrix M (also called *Schatten-1 norm*, *nuclear norm* or *trace norm* of M) and $\|M\|_{\mathcal{S}(\infty)}$ as the largest singular value of M (also called *spectral norm* of M).

(iii) $\|\mathbb{E}_{x\sim\mathcal{A}_{LM}}[(x-y)(x-y)^T]\|_{\mathcal{S}(\infty)} \leq O(1)$.

Proof (i) follows because our random walk and hence $x-y$ is orthogonal to $g_1, g_2, \ldots, g_m$. (ii) is explicitly in Lovett and Meka (2012). For (iii), we need to show that for any unit vector $\theta \in \mathbb{R}^n$, we have $\mathbb{E}_{x\sim\mathcal{A}_{LM}}[\langle\theta, x-y\rangle^2] \leq O(\|\theta\|_2^2)$. We use that the outcome of the random walk is of the form

$$x = y + \delta\sum_{t=1}^{O(1/\delta^2)} u_t \quad \text{where} \quad u_t \sim N(\mathbf{0}, \Sigma_t)$$

Here $0 \preceq \Sigma_t \preceq I_n$. But crucially each covariance matrix Σ_t may depend on the outcome of $u_1, \ldots, u_{t-1}$. In particular it is *not* true that $x-y$ is Gaussian. But it is a Martingale and as for each step t one has $\mathbb{E}[\langle u_t, \theta\rangle] = 0$ and $\mathbb{E}[\langle u_t, \theta\rangle^2] \leq O(\|\theta\|_2^2)$, the variance still satisfies $\mathbb{E}[\langle\delta\sum_{t=1}^{O(1/\delta^2)} u_t, \theta\rangle^2] \leq O(\|\theta\|_2^2)$ which settles (iii). Note that (iii) also follows from Lemma 10 in Lovett and Meka (2012), which posits that $x-y$ is subgaussian, implying the bound on the spectral norm of the covariance matrix stated in (iii). ■

2.2. Proof of Theorem 2

We are given a weight vector $y \in [0,1]^n$ and have access to samples $g_1, \ldots, g_m \sim \mathcal{D}_{\text{grad}}$ where $\mathcal{D}_{\text{grad}}$ is the gradient distribution on $\mathbb{R}^n$ whose covariance matrix $\Sigma := \mathbb{E}_{g\sim\mathcal{D}_{\text{grad}}}[gg^T]$ has rapidly decaying Eigenvalues, say $\lambda_k \leq \frac{\lambda_1}{k^\alpha}$ for some $\alpha > 1$. The algorithm to compute $\hat{x}$ as in Theorem 2 is simple:

```
LOVETT-MEKA ROUNDING ALGORITHM
Input: Weight vector $y \in [0,1]^n$ and parameter $m$
Output: Rounded vector $\hat{x}$
  (1) Sample $g_1, \ldots, g_m \sim \mathcal{D}_{\text{grad}}$. Initialize $x^{(0)} := y$
  (2) FOR $t = 0$ TO $\infty$ DO
      (3) IF $|\text{frac}(x^{(t)})| \leq 16m$ then return $\hat{x} = x^{(t)}$
      (4) Set $x^{(t+1)} := \mathcal{A}_{LM}(g_1, \ldots, g_m, x^{(t)})$
```

Let $\widehat{\Sigma} = \mathbb{E}_{i\in[m]}[g_i g_i^T]$ be the empirical covariance matrix. Note that by Theorem 3, each update $x^{(t)} - x^{(t-1)}$ is orthogonal to $g_1, \ldots, g_m$. Therefore $\hat{x} - y$ is also orthogonal to $g_1, \ldots, g_m$ and so $\widehat{\Sigma}(\hat{x} - y) = 0$, i.e., the empirical quantization error is zero. Therefore we now only need to bound the generalization error. We now do that as follows:

$$\begin{aligned}
\mathbb{E}_{\hat{x}}\mathbb{E}_{g\sim\mathcal{D}_{\text{grad}}}[\langle g, \hat{x}-y\rangle^2] &= \mathbb{E}_{\hat{x}}[(\hat{x}-y)^T\Sigma(\hat{x}-y)] \\
&= \underbrace{\mathbb{E}_{\hat{x}}[(\hat{x}-y)^T(\Sigma-\widehat{\Sigma})(\hat{x}-y)]}_{\text{generalization error}} + \underbrace{\mathbb{E}_{\hat{x}}[(\hat{x}-y)^T\widehat{\Sigma}(\hat{x}-y)]}_{\text{empirical error}=0} \\
&= \left\langle \Sigma - \widehat{\Sigma}, \mathbb{E}_{\hat{x}}[(\hat{x}-y)(\hat{x}-y)^T]\right\rangle \\
&\leq \|\Sigma-\widehat{\Sigma}\|_{S(1)} \cdot \|\mathbb{E}_{\hat{x}}[(\hat{x}-y)(\hat{x}-y)^T]\|_{S(\infty)}
\end{aligned}$$

where $\|\cdot\|_{S(1)}$ is the trace norm and $\|\cdot\|_{S(\infty)}$ is the spectral norm respectively and the last step follows from Matrix Hölder's inequality. A crucial aspect of analyzing this algorithm is understanding how far the empirical covariance $\widehat{\Sigma}$ is from the actual covariance matrix Σ in terms of the trace norm $\|\cdot\|_{\mathcal{S}(1)}$. We prove the following proposition Appendix B. We note that although prior work has shown bounds on the Schatten-∞ (spectral) norm of the covariance estimator, our bound on the Schatten-1 (trace) norm is novel and non-trivial. Importantly, it is the crucial step where we invoke the decay of the eigenvalues.

Proposition 4 *Let $\alpha > 1$, $\beta \geq 1$ and let $\mathcal{D}$ be a β-reasonable distribution with covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$ whose Eigenvalues satisfy $\lambda_k \leq \frac{C}{k^\alpha}$ for all $k = 1, \ldots, n$. Let $g_1, \ldots, g_m \sim \mathcal{D}$ be independent samples and let $\widehat{\Sigma} = \mathbb{E}_{i\in[m]}[g_i g_i^T]$ be the empirical covariance matrix. Then*

$$\mathbb{E}[\|X - \Sigma\|_{\mathcal{S}(1)}] \lesssim_{\alpha,\beta} C \cdot F_\alpha(m, n)$$

where $F_\alpha(m,n)$ is as defined in Theorem 2.

Now we are left with bounding $\|\mathbb{E}_{\hat{x}}[(\hat{x}-y)(\hat{x}-y)^T]\|_{S(\infty)}$ and to prove that $\text{frac}(\hat{x}) = O(m)$. Here we will use (ii) and (iii) from Theorem 3. Suppose $x^{(t^*)}$ is the vector that the algorithm returned in (3). It will be notationally convenient to define $x^{(t)} := x^{(t^*)}$ for all $t > t^*$. We say that iteration t is *good* if either $|\text{frac}(x^{(t-1)})| \leq 16m$ or if $|\text{frac}(x^{(t)})| \leq \frac{1}{2}|\text{frac}(x^{(t-1)})|$. If an iteration t is not good, we repeat the iteration until it is good. From Theorem 3.(ii) we know that every iteration is good with probability at least $\frac{1}{10}$ (independently of previous outcomes), thus by standard Chernov bounds, with probability at least 0.99, within the first $T := C' \log(\frac{n}{m})$ iterations there must be at least $\log(\frac{n}{m})$ many good iterations, for $C' > 0$ a sufficiently large constant. After $\log(\frac{n}{m})$ good iterations, one has $|\text{frac}(x^{(T)})| \leq 16m$. We have $\hat{x} - y = \sum_{t=0}^{T-1}(x^{(t+1)} - x^{(t)})$. Note that the update $(x^{(t+1)} - x^{(t)})$ in iteration t has zero mean conditioned on the previous updates, so $\mathbb{E}[(x^{(t+1)} - x^{(t)})(x^{(t'+1)} - x^{(t')})^T] = 0$ for $t \neq t'$. Therefore

$$\begin{aligned}
\|\mathbb{E}[(\hat{x}-y)(\hat{x}-y)^T]\|_{S(\infty)} &= \left\|\mathbb{E}\left[\sum_{t=0}^{T-1}(x^{(t+1)} - x^{(t)})(x^{(t+1)} - x^{(t)})^T\right]\right\|_{S(\infty)} \\
&\leq \sum_{t=0}^{T-1}\left\|\mathbb{E}\left[(x^{(t+1)} - x^{(t)})(x^{(t+1)} - x^{(t)})^T\right]\right\|_{S(\infty)} \\
&\leq O(T). \qquad \text{(By Theorem 3.(iii))}
\end{aligned}$$

This completes the proof of Theorem 2.

3. DiscQuant: Algorithm

In this section, we will present *DiscQuant*, a simple and practical algorithm for rounding inspired by the theoretical insights in Section 2. Instead of trying to approximate the loss function of the pre-trained model, i.e., $f(\hat{w}; s) \approx f(w; s)$, we will instead take a distillation approach and try to minimize the KL divergence between the next token distribution of the original model and the quantized model. Let $p_w(\cdot|z_{<i})$ be the distribution of the next token predicted by the original model given prefix $z_{<i}$ where $z \sim \mathcal{D}_{\text{data}}$ is a sample from the data distribution. We want $\text{error}(\hat{w}) = \mathbb{E}_{z\sim\mathcal{D}_{\text{data}}}\mathbb{E}_i D_{KL}\left(p_w(\cdot|z_{<i}) \parallel p_{\hat{w}}(\cdot|z_{<i})\right) \approx 0$. Expanding the KL divergence using Taylor series, we

can see that first order term vanishes exactly and so the second order term is the dominant term (see Appendix D). By Lemma 5,

$$D_{KL}\left(p_w(\cdot|z_{<i}) \parallel p_{\hat{w}}(\cdot|z_{<i})\right) \approx \mathbb{E}_{t\sim p_w(\cdot|z_{<i})}\left[\langle \nabla_w \log p_w(t|z_{<i}), \hat{w}-w\rangle^2\right].$$

So minimizing KL divergence is a succinct way to impose constraints of the form $\langle \nabla_w \log p_w(t|z_{<i}), \hat{w}- w\rangle \approx 0$ or equivalently $\log p_w(t|z_{<i}) \approx \log p_{\hat{w}}(t|z_{<i})$ where $t \sim p_w(\cdot|z_{<i})$ and $z \sim \mathcal{D}_{\text{data}}$. Therefore, we can use the same techniques developed in Section 2 to solve this as well. Assuming that the gradients are low rank, the set of x satisfying these constraints (where $\hat{w} = w^x$) form an affine subspace V of dimension $\geq n - m$ where m is the number of samples. We are again interested in finding a vertex of the polytope $K = [0,1]^n \cap V$ which will have $\geq n - m$ integral coordinates. At this point, we could use the Lovett-Meka algorithm (Algorithm 2.2) which has provable generalization guarantees. But explicitly calculating all the gradients and storing them is infeasible. Instead a simple heuristic way to find a random vertex of polytope K is to minimize a random linear function. This is inspired by the Eldan and Singh (2018) algorithm for discrepancy minimization where they pick a random linear function and optimize it over the $K \cap [-1,1]^n$ polytope to find a good rounding. However the rounding point given by Eldan and Singh does not satisfy the subgaussian bound (part (iii) of Theorem 3), therefore we cannot use it directly in our theory. But in practice this is a much more efficient way to find a good rounding point. We can approximately say that the DiscQuant algorithm is applying the Eldan and Singh algorithm instead of Lovett-Meka.

Let $c \in \mathbb{R}^n$ be some arbitrary vector; we will try to minimize the linear function $\langle c, x\rangle$ along with the KL divergence by taking a linear combination of them. The final optimization objective is shown in (4) where $\lambda > 0$ is a regularization coefficient.

$$\begin{aligned} &\min_x \lambda \langle c, x\rangle + \mathbb{E}_{z\sim\mathcal{D}_{\text{data}}}\mathbb{E}_i[D_{KL}\left(p_w(\cdot|z_{<i}) \parallel p_{w^x}(\cdot|z_{<i})\right)] \\ &s.t.\ x \in [0,1]^n. \end{aligned} \tag{4}$$

We solve the optimization problem (4) using projected stochastic gradient descent where we project x to the hypercube after every gradient update. Optimizing (4) will keep us close the polytope K and will approximately converge to a vertex of K which is almost integral. We round whatever fractional coordinates are left using RTN to get a fully integral solution.

We use one additional heuristic to improve the performance of the algorithm in practice. Instead of choosing a random vertex of the polytope K by choosing the vector c at random, we will choose it carefully so as to find the vertex of the polytope K which is closest to y which is the interpolation point corresponding to the original model weights (i.e., y such that $w^y = w$). We have:

$$\|x-y\|^2 = \sum_i (x_i^2 - 2x_iy_i + y_i^2) \approx \sum_i (x_i - 2x_iy_i + y_i^2) = \langle c^*, x\rangle + \|y\|^2$$

where $c^* = (1-2y)$. Here we have used the fact that $x_i^2 = x_i$ whenever $x_i \in \{0,1\}$ and since x is almost integral, we can use the approximation in the summation above. With this approximation, minimizing $\|x-y\|^2$ over almost integral x is equivalent to minimizing $\langle c^*, x\rangle$. So in the DiscQuant algorithm, we use $c = c^*$ specifically instead of a random c.

4. Experiments

We start by noting that our goal is not to achieve SOTA quantization performance, but to show that "careful" rounding via DiscQuant can improve the performance for any quantization format.

Method	Wbits	Wiki↓	GSM8k↑	MMLU↑	ArcC↑	PIQA↑	Hella↑	Wino↑
—	16.0	9.5	84.4$_{\pm 1.0}$	70.4$_{\pm 0.4}$	56.7$_{\pm 1.4}$	80.8$_{\pm 0.9}$	77.4$_{\pm 0.4}$	73.5$_{\pm 1.2}$
RTN	3.0	6.3E5	1.0$_{\pm 0.3}$	23.3$_{\pm 0.4}$	26.9$_{\pm 1.3}$	53.4$_{\pm 1.2}$	28.2$_{\pm 0.4}$	48.6$_{\pm 1.4}$
GPTQ	3.0	28.2	2.3$_{\pm 0.4}$	37.7$_{\pm 0.4}$	34.8$_{\pm 1.4}$	64.3$_{\pm 1.1}$	56.5$_{\pm 0.5}$	52.6$_{\pm 1.4}$
DiscQ	3.0	17.7	26.8$_{\pm 1.2}$	45.6$_{\pm 0.4}$	44.1$_{\pm 1.5}$	73.9$_{\pm 1.0}$	63.3$_{\pm 0.5}$	66.6$_{\pm 1.3}$
RTN	3.25	22.5	31.0$_{\pm 1.3}$	53.2$_{\pm 0.4}$	48.4$_{\pm 1.5}$	72.5$_{\pm 1.0}$	68.3$_{\pm 0.5}$	62.6$_{\pm 1.4}$
GPTQ	3.25	13.8	54.3$_{\pm 1.4}$	59.0$_{\pm 0.4}$	49.6$_{\pm 1.5}$	77.3$_{\pm 1.0}$	71.1$_{\pm 0.5}$	66.5$_{\pm 1.3}$
DiscQ	3.25	12.6	64.2$_{\pm 1.3}$	60.7$_{\pm 0.4}$	53.5$_{\pm 1.5}$	78.7$_{\pm 1.0}$	72.3$_{\pm 0.4}$	72.5$_{\pm 1.3}$
RTN	3.5	18.8	46.3$_{\pm 1.4}$	57.0$_{\pm 0.4}$	46.2$_{\pm 1.5}$	73.8$_{\pm 1.0}$	70.0$_{\pm 0.5}$	63.9$_{\pm 1.4}$
GPTQ	3.5	12.8	54.6$_{\pm 1.4}$	61.7$_{\pm 0.4}$	51.6$_{\pm 1.5}$	78.9$_{\pm 1.0}$	72.3$_{\pm 0.4}$	68.3$_{\pm 1.3}$
DiscQ	3.5	12.0	69.5$_{\pm 1.3}$	63.0$_{\pm 0.4}$	51.1$_{\pm 1.5}$	78.9$_{\pm 1.0}$	73.0$_{\pm 0.4}$	73.9$_{\pm 1.2}$
RTN	4.0	14.6	62.2$_{\pm 1.3}$	61.2$_{\pm 0.4}$	53.6$_{\pm 1.5}$	76.3$_{\pm 1.0}$	72.9$_{\pm 0.4}$	65.3$_{\pm 1.3}$
GPTQ	4.0	11.5	71.5$_{\pm 1.2}$	65.1$_{\pm 0.4}$	54.6$_{\pm 1.5}$	78.8$_{\pm 1.0}$	74.7$_{\pm 0.4}$	70.9$_{\pm 1.3}$
DiscQ	4.0	11.2	77.3$_{\pm 1.2}$	65.7$_{\pm 0.4}$	56.8$_{\pm 1.4}$	79.5$_{\pm 0.9}$	74.5$_{\pm 0.4}$	72.0$_{\pm 1.3}$
RTN	4.25	11.2	64.4$_{\pm 1.3}$	67.5$_{\pm 0.4}$	55.5$_{\pm 1.5}$	79.3$_{\pm 0.9}$	76.1$_{\pm 0.4}$	69.1$_{\pm 1.3}$
GPTQ	4.25	10.3	81.0$_{\pm 1.1}$	68.5$_{\pm 0.4}$	56.9$_{\pm 1.4}$	79.7$_{\pm 0.9}$	76.1$_{\pm 0.4}$	72.1$_{\pm 1.3}$
DiscQ	4.25	10.2	80.7$_{\pm 1.1}$	68.4$_{\pm 0.4}$	57.3$_{\pm 1.4}$	80.7$_{\pm 0.9}$	76.3$_{\pm 0.4}$	74.2$_{\pm 1.2}$
RTN	4.5	10.8	71.6$_{\pm 1.2}$	67.7$_{\pm 0.4}$	57.5$_{\pm 1.4}$	79.3$_{\pm 0.9}$	76.6$_{\pm 0.4}$	72.2$_{\pm 1.3}$
GPTQ	4.5	10.1	82.0$_{\pm 1.1}$	68.8$_{\pm 0.4}$	55.8$_{\pm 1.5}$	80.8$_{\pm 0.9}$	76.5$_{\pm 0.4}$	71.8$_{\pm 1.3}$
DiscQ	4.5	10.0	82.1$_{\pm 1.1}$	68.5$_{\pm 0.4}$	56.6$_{\pm 1.4}$	80.2$_{\pm 0.9}$	76.7$_{\pm 0.4}$	74.2$_{\pm 1.2}$

Table 2: Phi-3-mini-4k-instruct. Across all tasks and bits, our method DiscQuant always achieves superior results over the baseline RTN and GPTQ methods. On the ArcC, PIQA, and Wino tasks, DiscQuant achieves full recovery with at least 0.25 fewer bits per parameter than GPTQ and RTN.

Since GPTQ and RTN are the only rounding methods used in SOTA works on quantization, we only compare against them but on a fixed and simple quantization format like block scaling.

We evaluate our method on the Phi-3-mini-4k-instruct (Abdin et al., 2024) and Meta-Llama-3.1-8B-Instruct (Dubey et al., 2024) models, and compare against GPTQ and RTN. We use the lm-evaluation-harness (Gao et al., 2023) to evaluate on the Wikitext, GSM8k_cot 8-shot, MMLU 5-shot, ARC_Challenge 0-shot, PIQA 0-shot, HellaSwag 0-shot, and Winogrande 0-shot tasks. We report standard errors from lm-evaluation-harness. Wikitext measures perplexity, GSM8k is a generative task, and the remaining are multiple choice tasks. Note that generative tasks are typically more difficult than multiple choice tasks, and better reflect how the models are used in practice. See Appendix E for details on the hardware used, and hyper-parameter settings. Our method has similar memory requires as knowledge distillation, which also requires two copies of the model. We do not perform inference timing experiments; DiscQuant can optimize over a given quantization grid, so that we can utilize any pre-existing inference optimizations. For example, there are inference kernels for block scaling (Frantar et al., 2024) and incoherence processing (Tseng et al., 2024a). Ablations on the loss formulation are in Appendix E.

4.1. Block Scaling

We use standard block scaling quantization, determined by a `bits` and `groupsize` parameter. There are $2^{\texttt{bits}}$ unique points, and every `groupsize` parameters share a unique 16-bit scale pa-

Method	Wbits	Wiki↓	GSM8k↑	MMLU↑	ArcC↑	PIQA↑	Hella↑	Wino↑
—	16.0	8.7	77.0$_{\pm 1.2}$	68.0$_{\pm 0.4}$	55.2$_{\pm 1.5}$	81.3$_{\pm 0.9}$	79.3$_{\pm 0.4}$	73.7$_{\pm 1.2}$
RTN	3.0	4.4E3	0.5$_{\pm 0.2}$	23.2$_{\pm 0.4}$	22.3$_{\pm 1.2}$	52.4$_{\pm 1.2}$	29.1$_{\pm 0.5}$	50.0$_{\pm 1.4}$
GPTQ	3.0	23.2	3.6$_{\pm 0.5}$	24.6$_{\pm 0.4}$	31.8$_{\pm 1.4}$	66.6$_{\pm 1.1}$	45.8$_{\pm 0.5}$	54.1$_{\pm 1.4}$
DiscQ	3.0	15.2	14.3$_{\pm 1.0}$	44.6$_{\pm 0.4}$	39.4$_{\pm 1.4}$	73.2$_{\pm 1.0}$	64.4$_{\pm 0.5}$	62.8$_{\pm 1.4}$
RTN	3.25	15.2	10.8$_{\pm 0.9}$	50.5$_{\pm 0.4}$	44.3$_{\pm 1.5}$	75.2$_{\pm 1.0}$	71.4$_{\pm 0.5}$	67.2$_{\pm 1.3}$
GPTQ	3.25	10.7	56.3$_{\pm 1.4}$	60.5$_{\pm 0.4}$	46.3$_{\pm 1.5}$	76.7$_{\pm 1.0}$	74.4$_{\pm 0.4}$	68.7$_{\pm 1.3}$
DiscQ	3.25	10.5	58.3$_{\pm 1.4}$	60.2$_{\pm 0.4}$	49.1$_{\pm 1.5}$	79.1$_{\pm 0.9}$	75.1$_{\pm 0.4}$	72.1$_{\pm 1.3}$
RTN	3.5	12.7	35.9$_{\pm 1.3}$	51.4$_{\pm 0.4}$	48.4$_{\pm 1.5}$	76.7$_{\pm 1.0}$	73.0$_{\pm 0.4}$	69.1$_{\pm 1.3}$
GPTQ	3.5	10.4	57.0$_{\pm 1.4}$	62.1$_{\pm 0.4}$	49.9$_{\pm 1.5}$	77.3$_{\pm 1.0}$	75.1$_{\pm 0.4}$	71.1$_{\pm 1.3}$
DiscQ	3.5	10.3	60.7$_{\pm 1.3}$	60.9$_{\pm 0.4}$	51.7$_{\pm 1.5}$	79.2$_{\pm 0.9}$	76.3$_{\pm 0.4}$	72.5$_{\pm 1.3}$
RTN	4.0	12.5	50.8$_{\pm 1.4}$	59.3$_{\pm 0.4}$	50.5$_{\pm 1.5}$	77.6$_{\pm 1.0}$	74.7$_{\pm 0.4}$	69.9$_{\pm 1.3}$
GPTQ	4.0	9.9	63.2$_{\pm 1.3}$	64.4$_{\pm 0.4}$	52.4$_{\pm 1.5}$	78.4$_{\pm 1.0}$	75.9$_{\pm 0.4}$	71.7$_{\pm 1.3}$
DiscQ	4.0	9.8	66.5$_{\pm 1.3}$	63.4$_{\pm 0.4}$	51.6$_{\pm 1.5}$	79.2$_{\pm 0.9}$	76.9$_{\pm 0.4}$	72.8$_{\pm 1.3}$
RTN	4.25	9.4	70.6$_{\pm 1.3}$	65.7$_{\pm 0.4}$	54.2$_{\pm 1.5}$	80.1$_{\pm 0.9}$	78.0$_{\pm 0.4}$	73.9$_{\pm 1.2}$
GPTQ	4.25	9.1	74.6$_{\pm 1.2}$	66.8$_{\pm 0.4}$	53.4$_{\pm 1.5}$	79.6$_{\pm 0.9}$	77.9$_{\pm 0.4}$	73.5$_{\pm 1.2}$
DiscQ	4.25	9.1	74.9$_{\pm 1.2}$	66.9$_{\pm 0.4}$	53.6$_{\pm 1.5}$	79.9$_{\pm 0.9}$	78.4$_{\pm 0.4}$	72.6$_{\pm 1.3}$
RTN	4.5	9.3	71.9$_{\pm 1.2}$	65.8$_{\pm 0.4}$	54.8$_{\pm 1.5}$	80.3$_{\pm 0.9}$	78.4$_{\pm 0.4}$	72.4$_{\pm 1.3}$
GPTQ	4.5	9.0	73.8$_{\pm 1.2}$	66.9$_{\pm 0.4}$	53.6$_{\pm 1.5}$	79.6$_{\pm 0.9}$	78.1$_{\pm 0.4}$	73.7$_{\pm 1.2}$
DiscQ	4.5	9.1	74.8$_{\pm 1.2}$	66.8$_{\pm 0.4}$	54.1$_{\pm 1.5}$	80.6$_{\pm 0.9}$	78.7$_{\pm 0.4}$	72.9$_{\pm 1.2}$

Table 3: Meta-Llama-3.1-8B-Instruct. Our method DiscQuant achieves superior compression on the vast majority of quantization levels and tasks over the baselines GPTQ and RTN.

rameter. For example, 3.25 bits is achieved with `bits=3, groupsize=64`. We use the block scaling implementation from Frantar et al. (2024) which is symmetric linear quantization. Table 2 shows the results quantizing Phi-3-mini-4k-instruct. Across all tasks and all bit settings, our method DiscQuant achieves superior or comparable compression over the baseline GPTQ and RTN methods. The gap between DiscQuant and the baselines is greater at lower bits. On the ARC_Challenge, PIQA, and WinoGrade tasks, DiscQuant achieves full recovery with at least 0.25 fewer bits per parameter than GPTQ and RTN. DiscQuant achieves better compression on the more difficult generative GSM8k task: at 4 bits DiscQuant gets 77.3% accuracy, while GPTQ gets 71.5%, and RTN gets 62.2%. Table 3 shows the results quantizing Meta-Llama-3.1-8B-Instruct. Our method DiscQuant achieves improved compression on the majority of quantization levels and tasks.

4.2. Incoherence Processing

We explore another quantization format to show that our method can compose with other methods. Incoherence processing has been shown to improve quantization, especially at less than 4 bits per weight (Chee et al., 2023). The weights are multiplied by certain random orthogonal matrices prior to quantization, which can reduce the range of the weights and make quantization easier. We employ the Randomized Hadamard Transform from Tseng et al. (2024a). We use the same block scaling quantization grid as in the previous subsection. An abbreviated set of results are shown in Figure 5, where we superimpose bar plots for block scaling and block scaling + incoherence processing. In the majority of cases, adding incoherence processing increases the task accuracy, especially at lower

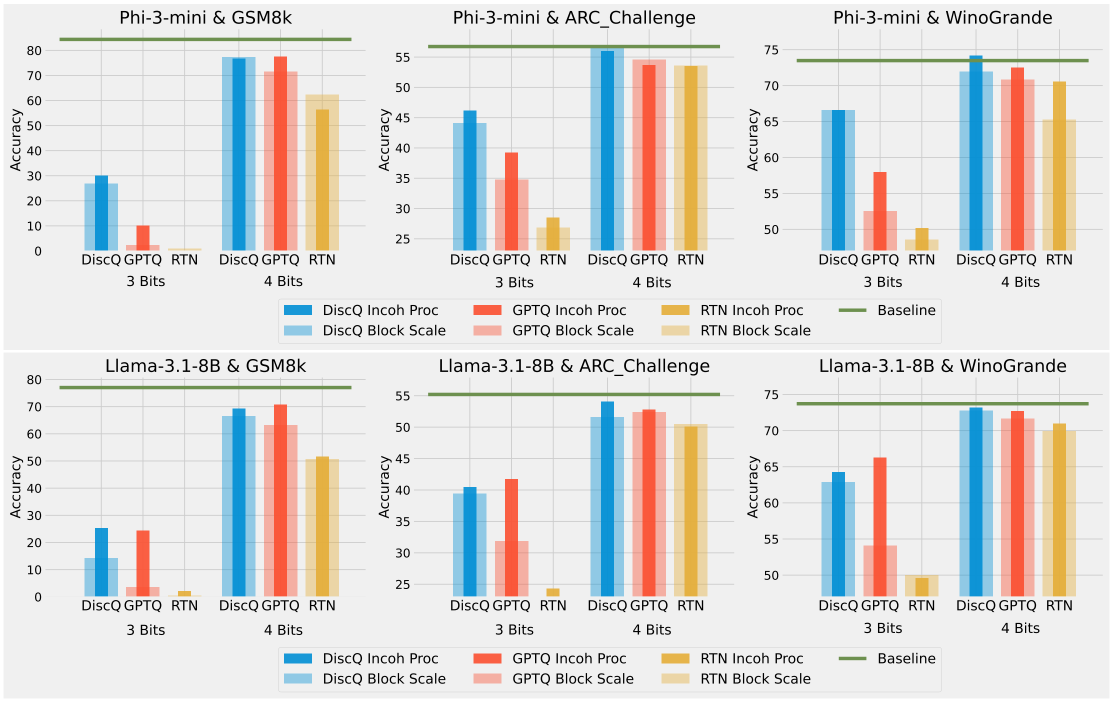

Figure 5: Quantizing Phi-3-mini-4k-instruct and Meta-LLama-3.1-8B-Instruct with block scaling, and additional incoherence processing. DiscQuant can compose with other quantization improvements, and with incoherence processing remains competitive with GPTQ.

bits. Incoherence especially helps GPTQ at 3 bits, and for Phi-3 DiscQuant without incoherence is competitive to GPTQ with incoherence. See Appendix E for the full set of results.

4.3. QAT

We run knowledge distillation with QAT in Table 4, using the same number of samples and training iterations as our method, and tune the learning rate. Our method DiscQuant is superior at both bit settings. QAT seems to degrade further at lower bits; it is significantly worse at the lower 3 bit setting.

Method	Wbits	Wiki↓	GSM8k↑	PIQA↑
Base Model	16.0	9.5	$84.4_{\pm 1.0}$	$80.8_{\pm 0.9}$
QAT	3.0	68.0	$2.3_{\pm 0.4}$	$60.6_{\pm 1.1}$
DiscQ	3.0	17.7	$26.8_{\pm 1.2}$	$73.9_{\pm 1.0}$
QAT	3.5	12.3	$66.3_{\pm 1.3}$	$78.2_{\pm 1.0}$
DiscQ	3.5	12.0	$69.5_{\pm 1.3}$	$78.9_{\pm 1.0}$

Table 4: QAT baseline on Phi-3-mini-4k-instruct. QAT seems to degrade further at lower bits; it is significantly worse at the lower 3 bit setting.

4.4. Additional Experiments

In Appendix E we show additional experiments on the effect of training data on DiscQuant, and some ablations. We also show some experiments on how to further improve performance of models quantized with DiscQuant by fine-tuning LoRA adapters on top.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, and Harkirat Behl. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In *Thirty-either Conference on Neural Information Processing Systems*, 2024.

Nikhil Bansal. Discrepancy theory and related algorithms. In *Proc. Int. Cong. Math*, volume 7, pages 5178–5210, 2022.

Kayhan Behdin, Ayan Acharya, Aman Gupta, Sathiya Keerthi, Rahul Mazumder, Zhu Siyu, and Song Qingquan. Quantease: Optimization-based quantization for language models–an efficient and intuitive algorithm. *arXiv preprint arXiv:2309.01885*, 2023.

Bernard Chazelle, William WL Chen, and Anand Srivastav. Discrepancy theory and its applications. *Oberwolfach Reports*, 1(1):673–722, 2004.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xrk9g5vcXR.

Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Forty-First International Conference on Machine Learning*, 2024.

Ronen Eldan and Mohit Singh. Efficient algorithms for discrepancy minimization in convex sets. *Random Structures & Algorithms*, 2018.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *Proceedings of the International Conference on Machine Learning*, 2023.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. *arXiv preprint arXiv:2408.11743*, 2024.

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. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Babak Hassibi, Daivd G Stork, and Gregory J Wolff. optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, 1993.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Itay Hubara, Yury Nahshan, Yair Hanami, Ron Banner, and Daniel SOudry. Accurate post training quantization with small calibration sets. In *Thirty-Eighth International Conference on Machine Learning*, 2021.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. In *Forty-First International Conference on Machine Learning*, 2024.

Eldar Kurtic, Denis Kuznedelev, Elias Frantar, Michael Goin, and Dan Alistarh. Sparse fine-tuning for inference acceleration of large language models, 2023. URL https://arxiv.org/abs/2310.06927.

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

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

Yuang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *The Nineth International Conference on Learning Representations*, 2021.

Jin Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Acttivation-aware weight quantization for on-device llm compression and acceleration. In *Seventh Conference on Machine Learning and Systems*, 2024.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024a.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *Forty-First International Conference on Machine Learning*, 2024b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *The International Conference on Learning Representations*, 2019.

László Lovász, Joel Spencer, and Katalin Vesztergombi. Discrepancy of set-systems and matrices. *European Journal of Combinatorics*, 7(2):151–160, 1986.

Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. In *FOCS*, pages 61–67. IEEE Computer Society, 2012.

Eric Lybrand and Rayan Saab. A greedy algorithm for quantizing neural networks. *Journal of Machine Learning Research*, 22(156):1–38, 2021.

Jiri Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 2009.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7197–7206. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/nagel20a.html.

Pranav Ajit Nair and Arun Sai Suggala. Cdquant: Accurate post-training weight quantization of large pre-trained models using greedy coordinate descent, 2024. URL https://arxiv.org/abs/2406.17542.

Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023. URL https://openreview.net/forum?id=tz9JV2PRSv.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better llm quantization with hadamard incoherence and lattice codebooks. In *Forty-First International Conference on Machine Learning*, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems*, 2024b.

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

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

Appendix A. Related Work

In this paper we focus on weights-only PTQ. Quantization can also be applied to the activations or KV-cache (Ashkboos et al., 2024; Liu et al., 2024a,b). Other compression method such as pruning (Frantar and Alistarh, 2023; Sun et al., 2023) are also outside the scope of this work. As discussed in the introduction, post training quantization can be divided into two overall steps: (1) Construct a good low bit-complexity representations for the weights (the quantization grid), and (2) Round the original weights to the values in the quantization grid. To this date, the vast majority of PTQ research for LLMs has focused on step (1). Note that determining a good compressed representation can involve both encoding formats, as well as transformations to ensure the weights better match the encoding format.

A.1. Quantization Grids

One of the more common quantization formats is called block scaling, or group-wise quantization (Frantar et al., 2023). In addition to the `bits` parameter determining the number of representable points, each `groupsize` parameters share a unique scaling parameter. Another successful encoding is to identify a small set of important weights and keep them in high precision (Dettmers et al., 2022, 2024; Kim et al., 2024). Shao et al. (2024) learns quantization parameters. Other works apply transformations to make quantization easier, either relatively simple invariant scalings (Xiao et al., 2023; Lin et al., 2024), or more complicated random orthogonal transformations (Chee et al., 2023; Liu et al., 2024a). Beyond block scaling, there has been work quantizing multiple parameters together using vector quantization (Tseng et al., 2024a; Egiazarian et al., 2024; van Baalen et al., 2024) or trellis quantization (Tseng et al., 2024b).

A.2. Rounding

To the best of our knowledge, GPTQ (Frantar et al., 2023) is the main rounding method for LLMs. It is based on the Optimal Brain Surgeon (Hassibi et al., 1993), which was adapted for pruning and quantization in Frantar et al. (2022) and then refined for quantization in GPTQ. GPTQ works by minimizing a layer-wise objective $\|WX - \hat{W}X\|_2^2$, where W is the weight matrix of a linear layer and X is the matrix of input activations to that layer (stacked as columns). Chee et al. (2023) developed a rounding algorithm which they showed was equivalent to GPTQ. Two other LLM rounding methods both use coordinate descent: Nair and Suggala (2024) only has results on the closed source PaLM-2 models with no released code, and Behdin et al. (2023) has results on the OPT, BLOOM, and Falcon model families.

There was more work on rounding methods several years ago, before the LLM boom. These papers were typically on smaller vision models. The line of work was started by AdaRound (Nagel et al., 2020) and continuing to AdaQuant (Hubara et al., 2021) and BRECQ (Li et al., 2021) employ a similar approach to ours, optimizing essentially interpolation variables between the closest up(w^{up}) and down(w^{down}) quantization grid points, while adding a concave regularization term to encourage rounding and using a rectified sigmoid to interpolate between w^{up} and w^{down}. They also do rounding layer by layer. However our method uses a linear term as a regularizer inspired from our theoretical insights using discrepancy theory and uses simple linear interpolation between w^{up} and w^{down} and we round the entire model at once.

A.3. Discrepancy Theory

Discrepancy theory is a deep branch of mathematics and theoretical computer science, and we refer the readers to standard textbooks for more details (Matousek, 2009; Chazelle et al., 2004; Bansal, 2022) To our knowledge, only Lybrand and Saab (2021) makes the connection between discrepancy theory and quantization. However, besides the high level motivational similarities, their work is not directly relevant to ours. Lybrand and Saab (2021) reduce the problem of understanding the error introduced by quantization on the output of a single neuron to a problem in discrepancy, and construct an algorithm for quantizing a single neuron. Their theoretical analysis on the generalization error only applies to quantizing the first layer of a neural network. On the other hand, we use discrepancy theory to understand when the whole network $f(w; s)$ can be approximated by $f(\hat{w}; s)$ with $\hat{w}$ in the quantization grid, and our theory holds for any network as a whole as long as our assumptions are true.

Appendix B. Proof of Proposition 4: Analyzing the covariance estimator

Proof [Proof of Prop 4] WLOG we can assume that $C = 1$ by scaling. We first present the proof for the case of $1 < \alpha < \frac{3}{2}$ and then discuss the modifications for the other two cases. The claim is invariant under a change of basis, hence we may assume that Σ is a diagonal matrix with Eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n \geq 0$, i.e. $\Sigma_{ii} = \lambda_i$ for all $i \in [n]$. We can bound the variance terms for all entries (whether diagonal or not):
Claim I. *For all $i, j \in [n]$ one has $\mathbb{E}[|X_{ij} - \Sigma_{ij}|^2] \lesssim_\beta \frac{\lambda_i \lambda_j}{m}$.*
Proof of Claim I. We recall that $\mathbb{E}[X] = \Sigma$ and $\mathbb{E}[X^{(\ell)}] = \frac{1}{m}\Sigma$. For all $i, j \in [n]$ one has

$$\begin{aligned}
\mathbb{E}[|X_{ij} - \Sigma_{ij}|^2] &= \text{Var}[X_{ij}] \\
&= \sum_{\ell=1}^{m} \text{Var}[X_{ij}^{(\ell)}] \\
&= \frac{1}{m}\mathbb{E}_{h\sim\mathcal{D}}[|h_i h_j - \Sigma_{ij}|^2] \\
&\leq \frac{2}{m}\Big(\mathbb{E}_{h\sim\mathcal{D}}[h_i^2 h_j^2] + \underbrace{\Sigma_{ij}^2}_{\leq \lambda_i\lambda_j}\Big) \\
&\overset{(*)}{\leq} \frac{2}{m}(\mathbb{E}_{h\sim\mathcal{D}}[h_i^4]^{1/2}\mathbb{E}_{h\sim\mathcal{D}}[h_j^4]^{1/2} + \lambda_i\lambda_j) \\
&\overset{(**)}{\leq} \frac{2}{m}(\beta^{1/2}\underbrace{\mathbb{E}_{h\sim\mathcal{D}}[h_i^2]}_{=\lambda_i} \cdot \beta^{1/2}\underbrace{\mathbb{E}_{h\sim\mathcal{D}}[h_j^2]}_{=\lambda_j} + \lambda_i\lambda_j) = \frac{2\beta+2}{m}\cdot\lambda_i\lambda_j
\end{aligned}$$

Here we use the inequality $(a-b)^2 \leq 2a^2 + 2b^2$. Moveover $\Sigma_{ij} \leq \lambda_i \lambda_j$ holds because Σ is a diagonal matrix. Note that we have used Cauchy-Schwarz in $(*)$ and the assumption that $\mathcal{D}$ is β-reasonable in $(**)$. ■
Now let $J_\ell := \{i \in [n] \mid 2^{\ell-1} \leq i < 2^\ell\}$. It will be useful to note that $|J_\ell| \leq 2^\ell$ and the sum of the Eigenvalues in each block satisfies $\sum_{i\in J_\ell} \lambda_i \lesssim 2^\ell \cdot (2^\ell)^{-\alpha} = (2^\ell)^{1-\alpha}$. Our strategy is to use the triangle inequality to bound:

$$\mathbb{E}[\|X - \Sigma\|_{\mathcal{S}(1)}] \leq 2\sum_{\ell\geq 1}\sum_{k\geq \ell}\mathbb{E}[\|X_{J_\ell,J_k} - \Sigma_{J_\ell,J_k}\|_{\mathcal{S}(1)}] \tag{5}$$

Here X_{J_ℓ,J_k} is the $|J_\ell| \times |J_k|$ submatrix of X that is indexed by rows J_ℓ and columns J_k. In the following we will estimate the contribution of the different blocks depending on their parameter regime and whether they are diagonal or off-diagonal.

Claim II. *Let $\ell \leq k$ and abbreviate $Y := X_{J_\ell,J_k} - \Sigma_{J_\ell,J_k}$. Then*

$$\mathbb{E}[\|Y\|_{\mathcal{S}(1)}] \lesssim \sqrt{\frac{r}{m}} \cdot 2^{\frac{\ell+k}{2}(1-\alpha)}$$

assuming that rank$(Y) \leq r$ for any outcome of Y.

Proof of Claim II. We recall that for any matrix A one has $\|A\|_{\mathcal{S}(1)} \leq \sqrt{\text{rank}(A)} \cdot \|A\|_F$. Then for all $\ell \leq k$ we can bound

$$\begin{aligned}
\mathbb{E}[\|Y\|_{\mathcal{S}(1)}] &\leq \sqrt{r} \cdot \mathbb{E}[\|Y\|_F] \\
&\overset{\text{Jensen}}{\leq} \sqrt{r} \cdot \mathbb{E}[\|Y\|_F^2]^{1/2} \\
&\overset{\text{Claim I}}{\lesssim_\beta} \sqrt{r} \cdot \Big(\frac{1}{m}\Big(\sum_{i \in J_\ell} \lambda_i\Big)\Big(\sum_{j \in J_k} \lambda_j\Big)\Big)^{1/2} \\
&\lesssim \sqrt{r} \cdot \sqrt{\frac{1}{m} \cdot (2^\ell)^{1-\alpha} \cdot (2^k)^{1-\alpha}} \\
&= \sqrt{\frac{r}{m}} \cdot 2^{\frac{\ell+k}{2}(1-\alpha)}
\end{aligned}$$

■

Now we can bound the contribution that off-diagonal blocks have to Eq (5). Here we use that $\Sigma_{J_\ell,J_k} = \mathbf{0}$ and rank$(X_{J_\ell,J_k}) \leq \min\{m, 2^\ell\}$. Then

$$\begin{aligned}
\sum_{\ell \geq 1} \sum_{k > \ell} \mathbb{E}[\|X_{J_\ell,J_k} - \underbrace{\Sigma_{J_\ell,J_k}}_{=\mathbf{0}}\|_{\mathcal{S}(1)}] &\overset{\text{Claim II}}{\leq} \sum_{\ell \geq 1} \sum_{k > \ell} \frac{\sqrt{\min\{m, 2^\ell\}}}{\sqrt{m}} \cdot 2^{\frac{\ell+k}{2}(1-\alpha)} \\
&= \sum_{\ell \geq 1} \min\left\{1, \sqrt{2^\ell/m}\right\} \cdot 2^{\frac{\ell}{2}(1-\alpha)} \underbrace{\sum_{k > \ell} \cdot 2^{\frac{k}{2}(1-\alpha)}}_{\lesssim_\alpha 2^{\ell(1-\alpha)/2}} \\
&\lesssim_\alpha \sum_{\ell \geq 1} \min\left\{1, \sqrt{2^\ell/m}\right\} \cdot (2^\ell)^{1-\alpha} \qquad (6) \\
&\lesssim_\alpha m^{1-\alpha}
\end{aligned}$$

In the last step we use that the function $z \mapsto \sqrt{z} \cdot z^{1-\alpha}$ is monotonically increasing while $z \mapsto z^{1-\alpha}$ is monotonically decreasing as we assume that $1 < \alpha < \frac{3}{2}$. Hence the term with $m = 2^\ell$ dominates the sum.

It remains to bound the diagonal blocks. First we consider the regime of small indices. Here we use the bound rank$(X_{J_\ell,J_\ell} - \Sigma_{J_\ell,J_\ell}) \leq |J_\ell| \leq 2^\ell$ which gives

$$\sum_{\ell : 2^\ell \leq m} \mathbb{E}[\|X_{J_\ell,J_\ell} - \Sigma_{J_\ell,J_\ell}\|_{\mathcal{S}(1)}] \overset{\text{Claim II}}{\leq} \sum_{\ell : 2^\ell \leq m} \sqrt{\frac{2^\ell}{m}} \cdot 2^{\ell(1-\alpha)} \lesssim m^{1-\alpha} \qquad (7)$$

Here the last summand (with $2^\ell = m$) dominates the sum in (7), again as $z \mapsto \sqrt{z} \cdot z^{1-\alpha}$ is monotonically increasing.

The final regime to consider is the one of large indices, i.e. diagonal blocks with $2^\ell > m$. In that case we can ignore any concentration that the randomness may provide and simply bound

$$\begin{aligned}
\sum_{\ell:2^\ell>m} \mathbb{E}[\|X_{J_\ell,J_\ell} - \Sigma_{J_\ell,J_\ell}\|_{\mathcal{S}(1)}] &\leq \sum_{\ell:2^\ell>m} \left(\mathbb{E}[\|X_{J_\ell,J_\ell}\|_{\mathcal{S}(1)}] + \|\Sigma_{J_\ell,J_\ell}\|_{\mathcal{S}(1)}\right) \\
&= \sum_{\ell:2^\ell>m} \left(\mathbb{E}[\mathrm{Tr}[X_{J_\ell,J_\ell}]] + \mathrm{Tr}[\Sigma_{J_\ell,J_\ell}]\right) \\
&= \sum_{j=m}^{n} (\underbrace{\mathbb{E}[X_{jj}]}_{=\Sigma_{jj}} + \underbrace{\Sigma_{jj}}_{\leq j^{-\alpha}}) \\
&\lesssim \sum_{j\geq m} \frac{1}{j^\alpha} \lesssim m^{1-\alpha} \qquad (8)
\end{aligned}$$

Here we use again the triangle inequality of the trace norm and the fact that the matrices X_{J_ℓ,J_ℓ} and Σ_{J_ℓ,J_ℓ} are always positive semidefinite. This concludes the argument for $1 < \alpha < \frac{3}{2}$. If $\alpha = \frac{3}{2}$ then $\sqrt{2^\ell/m} \cdot (2^\ell)^{1-\alpha} \leq \frac{1}{\sqrt{m}}$ for each $\ell \geq 1$ and so (6) is bounded by $\frac{\log(n)}{\sqrt{m}}$. Moreover, the last two cases can be merged as

$$\sum_{\ell\geq 1} \mathbb{E}[\|X_{J_\ell,J_\ell} - \Sigma_{J_\ell,J_\ell}\|_{\mathcal{S}(1)}] \overset{\text{Claim II}}{\leq} \sum_{\ell\geq 1} \sqrt{\frac{2^\ell}{m}} \cdot 2^{\ell(1-\alpha)} \lesssim \frac{\log(n)}{\sqrt{m}} \qquad (9)$$

Finally, if $\alpha > \frac{3}{2}$ then the first term (for $\ell = 1$) dominates the sums in (6) and (9) and the extra $\log(n)$ term can be omitted. ■

Appendix C. Non-uniform Quantization Grid

We will introduce new parameters $x \in [0,1]^n$ and define

$$w^x = w^{\text{down}} \odot (1-x) + w^{\text{up}} \odot x$$

where $\odot$ is component-wise product. Note that w_i^x interpolates between w_i^{down} and w_i^{up} where $w_i = w_i^{\text{down}}$ if $x_i = 0$ and $w_i = w_i^{\text{up}}$ if $x_i = 1$. Let $y \in [0,1]^n$ be the interpolation point corresponding to the original weights, i.e., $w^y = w$. We can rewrite the linear constraints in terms of x as follows:

$$\begin{aligned}
\langle \nabla_w f(w;s_i), w^x - w\rangle &= \langle \nabla_w f(w;s_i), w^x - w^y\rangle \\
&= \left\langle \nabla_w f(w;s_i), (w^{\text{up}} - w^{\text{down}}) \odot (x-y)\right\rangle \\
&= \left\langle \nabla_w f(w;s_i) \odot (w^{\text{up}} - w^{\text{down}}), x-y\right\rangle.
\end{aligned}$$

Let M be an $m \times n$ matrix whose i^{th} row is given by $\nabla_w f(w;s_i) \odot (w^{\text{up}} - w^{\text{down}})$. Then the linear constraints can be simply written as $M(x-y) = 0$. We can prove a result similar to our main theorem (Theorem 2) by assuming that the covariance matrix of scaled gradients (i.e., $\nabla_w f(w;s_i) \odot (w^{\text{up}} - w^{\text{down}})$) has decaying eigenvalues.

Appendix D. Taylor Series for KL Divergence

Let $p_w(\cdot|z_{<i})$ be the distribution of the next token predicted by the original model given prefix $z_{<i}$ where $z \sim \mathcal{D}_{\text{data}}$ is a sample from the data distribution. Let

$$\text{error}(\hat{w}) = \mathbb{E}_{z\sim\mathcal{D}_{\text{data}}}\mathbb{E}_i D_{KL}\left(p_w(\cdot|z_{<i}) \parallel p_{\hat{w}}(\cdot|z_{<i})\right)$$

be the KL divergence between the original model and quantized model.

Lemma 5 *Let*

$$\textit{error}(\hat{w}) = \langle g_w, \hat{w} - w\rangle + (\hat{w} - w)^T H_w(\hat{w} - w) + \cdots$$

be the Taylor series expansion of the KL divergence where g_w is the gradient and H_w is the Hessian. Then

1. $g_w = 0$,
2. $H_w = \mathbb{E}_{z\sim\mathcal{D}_{\textit{data}}}\mathbb{E}_i\mathbb{E}_{t\sim p_w(\cdot|z_{<i})}[(\nabla_w \log p_w(t|z_{<i}))(\nabla_w \log p_w(t|z_{<i}))^T]$

Therefore $error(\hat{w}) \approx \mathbb{E}_{z\sim\mathcal{D}_{\textit{data}}}\mathbb{E}_i\mathbb{E}_{t\sim p_w(\cdot|z_{<i})}[\langle \nabla_w p_w(t|z_{<i}), \hat{w} - w\rangle^2]$.

Proof To simplify notation, we will ignore the z, i variables coming from $\mathbb{E}_{z\sim\mathcal{D}_{\text{data}}}$ and $\mathbb{E}_i$ and also drop them from $p_w(\cdot|z_{<i})$ and just write $p_w(\cdot)$. Adding these back and taking expectations over these variables, we get the desired result. We can expand the KL divergence using Taylor series and evaluate the first and second order terms.

$$\begin{aligned}
\text{error}(\hat{w}) &= D_{KL}\left(p_w(\cdot) \parallel p_{\hat{w}}(\cdot)\right) \\
&= -E_{t\sim}\left[\log p_{\hat{w}}(t) - \log p_w(t)\right] \\
&= -E_{t\sim p_w}\left[\langle \nabla_w \log p_w(t), \hat{w} - w\rangle + (\hat{w} - w)^T \nabla_w^2 \log p_w(t)(\hat{w} - w) + \cdots\right] \\
&= \langle g_w, \hat{w} - w\rangle + (\hat{w} - w)^T H_w(\hat{w} - w) + \cdots
\end{aligned}$$

where $g_w = -E_{t\sim p_w}[\nabla_w \log p_w(t)]$ and $H_w = -E_{t\sim p_w}[\nabla_w^2 \log p_w(t)]$.
(1) We first evaluate g_w.

$$\begin{aligned}
g_w = -E_{t\sim p_w}[\nabla_w \log p_w(t)] &= \mathbb{E}_{t\sim p_w}\left[\frac{\nabla_w p_w(t)}{p_w(t)}\right] \\
&= \sum_t \nabla_w p_w(t) \\
&= \nabla_w(\sum_t p_w(t)) \\
&= \nabla_w(1) = 0.
\end{aligned}$$

(2) We now evaluate H_w.

$$\begin{aligned}
H_w &= -E_{t\sim p_w}[\nabla_w^2 \log p_w(t)] \\
&= -\mathbb{E}_{t\sim p_w}\left[\nabla_w\left(\frac{\nabla_w p_w(t)}{p_w(t)}\right)\right] \\
&= -\mathbb{E}_{t\sim p_w}\left[\frac{\nabla_w^2 p_w(t)}{p_w(t)} - \frac{(\nabla_w p_w(t))(\nabla_w p_w(t))^T}{p_w(t)^2}\right] \\
&= -\mathbb{E}_{t\sim p_w}\left[\frac{\nabla_w^2 p_w(t)}{p_w(t)} - (\nabla_w \log p_w(t))(\nabla_w \log p_w(t))^T\right] \\
&= -\sum_t \nabla_w^2 p_w(t) + \mathbb{E}_{t\sim p_w}\left[(\nabla_w \log p_w(t))(\nabla_w \log p_w(t))^T\right] \\
&= -\nabla_w^2\left(\sum_t p_w(t)\right) + \mathbb{E}_{t\sim p_w}\left[(\nabla_w \log p_w(t))(\nabla_w \log p_w(t))^T\right] \\
&= -\nabla_w^2(1) + \mathbb{E}_{t\sim p_w}\left[(\nabla_w \log p_w(t))(\nabla_w \log p_w(t))^T\right] \\
&= \mathbb{E}_{t\sim p_w}\left[(\nabla_w \log p_w(t))(\nabla_w \log p_w(t))^T\right]
\end{aligned}$$

■

Appendix E. Additional Experiments

E.1. Experimental Setup Details

The experiments for the Phi-3-mini model were conducted on either a single 80GB Nvidia A100 GPU, or 2x40GB A100 GPUs, while the Llama-3.1-8B model used either 2x80GB A100s, or 4x40GB A100s. We use the PyTorch framework. We initialize $x \in [0,1]^n$ uniformly at random, and used AdamW (Loshchilov and Hutter, 2019) with a cosine learning rate schedule. We multiply the regularization coefficient λ with the KL loss term, and perform entry-wise gradient clipping on the KL loss term. For DiscQuant, we tuned the hyper-parameters for each model and bit setting. The hyper-parameters `clamp, λ, lr, batch_size, num_iter` and `warmup` were tuned. In the block scaling setting we found that `clamp={1.0, 0.5}, λ=200, lr={0.1, 0.05}, batch_size={4,8}, num_iter=1024, warmup=128` worked well for both models. In the incoherence processing setting we found that `clamp={0.05,0.01}, lr={0.05,0.01}` worked well for both models, all other parameters being the same as before. For GPTQ, we used the `actorder, true_sequential` heuristics, and tuned the number of samples over {`1024, 4096, 8192`} for each model and bit setting. Our quantization dataset is constructed from the RedPajama-1T-Sample training set (Computer, 2023). We concatenate random samples until up to 2048 sequence length, truncating the last sample if necessary. Greedy or round-to-nearest requires no data, and no hyper-parameter tuning.

E.2. Incoherence Processing

In Figures 6 and 7 we superimpose bar plots for block scaling and block scaling + incoherence processing. In the majority of cases, adding incoherence processing increases the task accuracy,

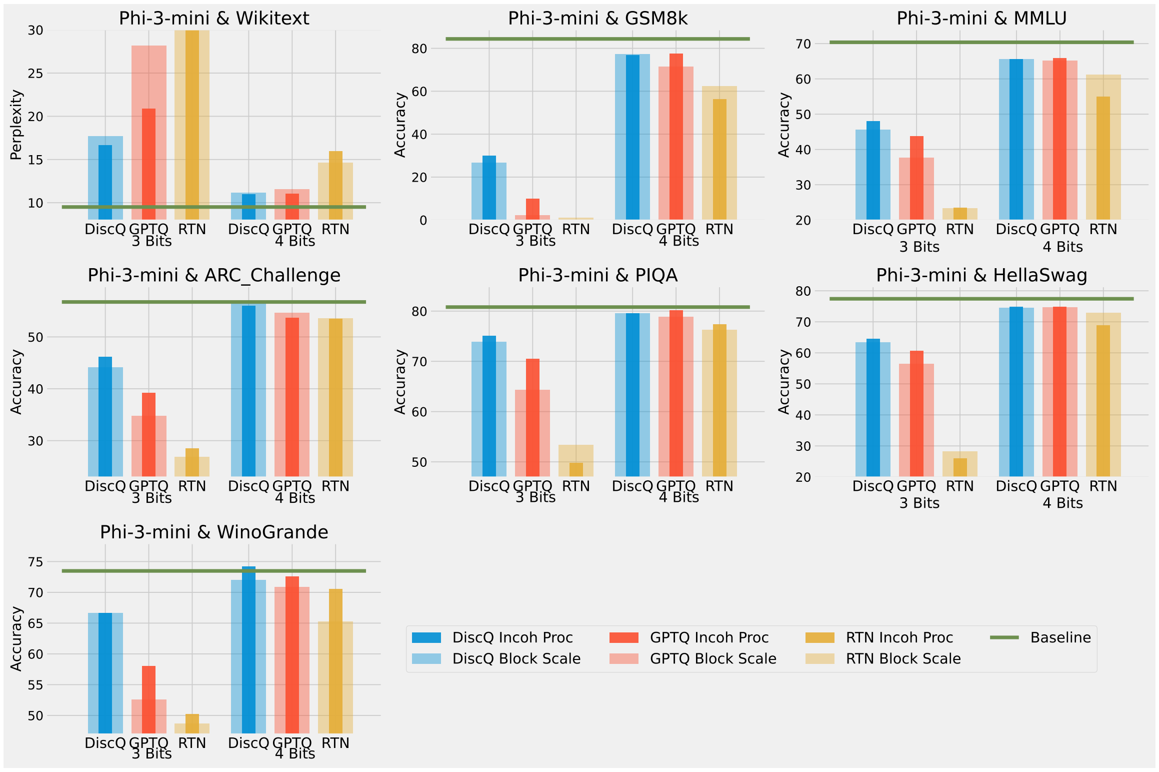

Figure 6: Quantizing Phi-3-mini-4k-instruct with block scaling, and additional incoherence processing. Adding incoherence processing largely improves model quality at 3 bits. At 4 bits, these improvements are smaller. At 3 bits, DiscQuant is better than GPTQ with incoherence processing.

Method	Wbits	Wiki↓	GSM8k↑	MMLU↑	ArcC↑	PIQA↑	Hella↑	Wino↑
—	16.0	9.5	$84.4_{\pm 1.0}$	$70.4_{\pm 0.4}$	$56.7_{\pm 1.4}$	$80.8_{\pm 0.9}$	$77.4_{\pm 0.4}$	$73.5_{\pm 1.2}$
RTN	3.0	2.6E5	$0.0_{\pm 0.0}$	$23.4_{\pm 0.4}$	$28.5_{\pm 1.3}$	$49.8_{\pm 1.2}$	$26.0_{\pm 0.4}$	$50.2_{\pm 1.4}$
GPTQ	3.0	20.8	$10.0_{\pm 0.8}$	$43.8_{\pm 0.4}$	$39.2_{\pm 1.4}$	$70.5_{\pm 1.1}$	$60.7_{\pm 0.5}$	$58.0_{\pm 1.4}$
DiscQ	3.0	16.7	$29.9_{\pm 1.3}$	$48.0_{\pm 0.4}$	$46.2_{\pm 1.5}$	$75.1_{\pm 1.0}$	$64.5_{\pm 0.5}$	$66.6_{\pm 1.3}$
RTN	4.0	15.9	$56.3_{\pm 1.4}$	$55.0_{\pm 0.4}$	$53.5_{\pm 1.5}$	$77.4_{\pm 1.0}$	$68.8_{\pm 0.5}$	$70.6_{\pm 1.3}$
GPTQ	4.0	11.0	$77.6_{\pm 1.1}$	$65.8_{\pm 0.4}$	$53.7_{\pm 1.5}$	$80.2_{\pm 0.9}$	$74.9_{\pm 0.4}$	$72.5_{\pm 1.3}$
DiscQ	4.0	11.0	$76.7_{\pm 1.2}$	$65.6_{\pm 0.4}$	$56.0_{\pm 1.5}$	$79.5_{\pm 0.9}$	$74.9_{\pm 0.4}$	$74.2_{\pm 1.2}$

Table 5: Phi-3-mini-4k-instruct with incoherence processing. At 3 bits per weight, DiscQuant achieves superior compression across all tasks. At 4 bits per weight, DiscQuant achieves comparable compression.

especially at lower bits. We do not use fractional bits, (i.e. no `groupsize`), due to the fact that both these methods effect outliers and can interfere with one another. Incoherence especially helps GPTQ at 3 bits, and for Phi-3 DiscQuant without incoherence is competitive to GPTQ with incoherence.

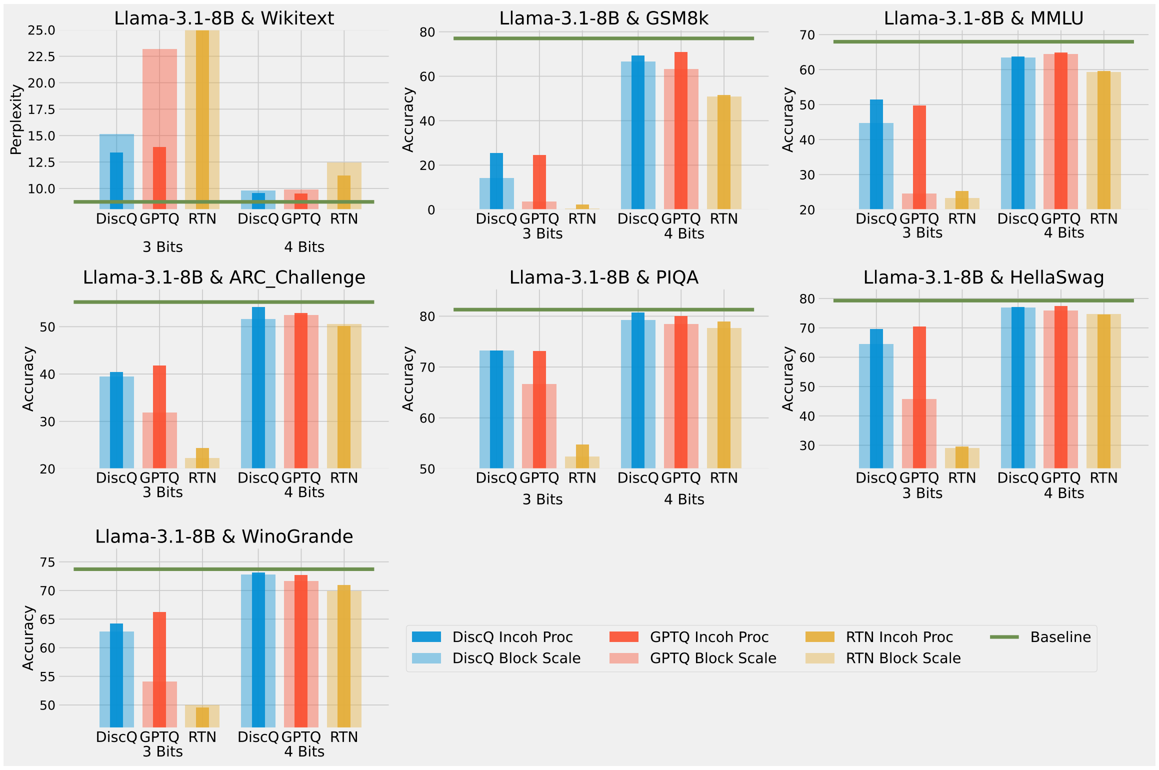

Figure 7: Quantizing Meta-Llama-3.1-8B-Instruct with block scaling, and additional incoherence processing. Adding incoherence processing largely improves model quality at 3 bits. At 4 bits, these improvements are smaller. After incoherence, DiscQuant is largely comparable to GPTQ.

Method	Wbits	Wiki↓	GSM8k↑	MMLU↑	ArcC↑	PIQA↑	Hella↑	Wino↑
—	16.0	8.7	77.0$\pm$1.2	68.0$\pm$0.4	55.2$\pm$1.5	81.3$\pm$0.9	79.3$\pm$0.4	73.7$\pm$1.2
RTN	3.0	2.4E3	2.1$\pm$0.4	25.2$\pm$0.4	24.3$\pm$1.3	54.7$\pm$1.2	29.5$\pm$0.5	49.6$\pm$1.4
GPTQ	3.0	13.9	24.4$\pm$1.2	49.7$\pm$0.4	41.7$\pm$1.4	73.1$\pm$1.0	70.4$\pm$0.5	66.2$\pm$1.3
DiscQ	3.0	13.4	25.4$\pm$1.2	51.5$\pm$0.4	40.4$\pm$1.4	73.2$\pm$1.0	69.6$\pm$0.5	64.2$\pm$1.3
RTN	4.0	11.2	51.6$\pm$1.4	59.5$\pm$0.4	50.1$\pm$1.5	78.9$\pm$1.0	74.5$\pm$0.4	71.0$\pm$1.3
GPTQ	4.0	9.5	70.7$\pm$1.3	64.9$\pm$0.4	52.8$\pm$1.5	80.0$\pm$0.9	77.4$\pm$0.4	72.7$\pm$1.3
DiscQ	4.0	9.6	69.4$\pm$1.3	63.7$\pm$0.4	54.1$\pm$1.5	80.7$\pm$0.9	77.0$\pm$0.4	73.2$\pm$1.2

Table 6: Meta-Llama-3.1-8B-Instruct with incoherence processing. Across a majority of bits and tasks, DiscQuant achieves comparable compression with GPTQ, and does better than RNT.

Table 5 shows our results quantizing Phi-3-mini-4k-instruct with incoherence processing. At 3 bits per weight, DiscQuant achieves superior compression across all tasks. At 4 bits per weight, DiscQuant achieves comparable compression. For example, on ARC_CHallenge at 3 bits, DiscQuant achieves 46.2% accuracy, while GPTQ achieves 39.2%, and RTN 28.5%. Table 6 shows our results quantizing Meta-Llama-3.1-8B-Instruct with incoherence processing. DiscQuant performs compa-

LoRA lr	LoRA rank	GSM8k↑	Wiki↓	MMLU↑	Wino↑
0.0	0	62.9$\pm$ 1.3	12.6	60.5$\pm$ 0.4	73.0$\pm$ 1.2
3E-06	8	63.2$\pm$ 1.3	12.6	60.8$\pm$ 0.4	72.8$\pm$ 1.3
3E-06	16	63.3$\pm$ 1.3	12.6	60.8$\pm$ 0.4	72.8$\pm$ 1.2
3E-06	32	63.3$\pm$ 1.3	12.6	60.8$\pm$ 0.4	73.0$\pm$ 1.2
1E-05	8	63.7$\pm$ 1.3	12.6	61.0$\pm$ 0.4	73.1$\pm$ 1.2
1E-05	16	63.5$\pm$ 1.3	12.6	60.9$\pm$ 0.4	73.2$\pm$ 1.2
1E-05	32	63.9$\pm$ 1.3	12.6	60.9$\pm$ 0.4	73.0$\pm$ 1.2
3E-05	8	64.4$\pm$ 1.3	12.5	61.1$\pm$ 0.4	73.0$\pm$ 1.2
3E-05	16	64.1$\pm$ 1.3	12.5	61.2$\pm$ 0.4	72.8$\pm$ 1.3
3E-05	32	64.1$\pm$ 1.3	12.5	61.0$\pm$ 0.4	72.9$\pm$ 1.2
1E-04	8	66.2$\pm$ 1.3	12.4	61.1$\pm$ 0.4	72.9$\pm$ 1.2
1E-04	16	66.7$\pm$ 1.3	12.4	61.4$\pm$ 0.4	72.6$\pm$ 1.3
1E-04	32	67.0$\pm$ 1.3	12.4	61.4$\pm$ 0.4	73.0$\pm$ 1.2
3E-04	8	65.3$\pm$ 1.3	12.3	61.2$\pm$ 0.4	73.5$\pm$ 1.2
3E-04	16	66.5$\pm$ 1.3	12.3	61.3$\pm$ 0.4	73.1$\pm$ 1.2
3E-04	32	66.8$\pm$ 1.3	12.3	61.4$\pm$ 0.4	73.1$\pm$ 1.2
1E-03	8	0.0$\pm$ 0.0	21056.1	22.9$\pm$ 0.4	51.3$\pm$ 1.4
1E-03	16	59.2$\pm$ 1.4	13.0	58.1$\pm$ 0.4	73.2$\pm$ 1.2
1E-03	32	59.4$\pm$ 1.4	12.9	58.5$\pm$ 0.4	72.7$\pm$ 1.3
		Base model			
—	—	84.4$\pm$ 1.0	9.5	70.4$\pm$ 0.4	73.5$\pm$ 1.2

Table 7: LoRA experiments with 3.25 bits. The first row corresponds to the optimal DiscQ model with 3.25 bits that is not trained with LoRA. The last row corresponds to the full precision model.

rably to GPTQ, and better than RTN. For example, on WinoGrande at 4 bits, DiscQuant achieves 73.2% accuracy, while GPTQ achieves 72.7%, and RTN 71.0%.

E.3. LoRA experiments

See Table 7 for our LoRA experiments. We initialize the model with the optimal choice of DiscQ for 3.25 bits and add LoRA adapters. We train LoRA adapters while freezing the rest of the parameters.

E.4. Effect of Data

We perform a simple investigation into the effect of the dataset on quantization. We mix math subject data–GSM8k and MetaMathQA–with our standard RedPajama dataset. In Figure 8 we give the full set of evaluation tasks when changing the mix of math subject data when quantizing Phi-3-mini-4k-instruct to 3.25 bits.

As expected, both methods increase accuracy on GSM8k when there is a greater fraction of math data. On HellaSwag, DiscQuant improves with more math data, where GPTQ gets worse. On PIQA, both methods get worse. It is interesting that across all evaluation tasks, there is a meaningful change in evaluation metrics as a result of changing the data mix. We leave the question of appropriate data curation as an important open question.

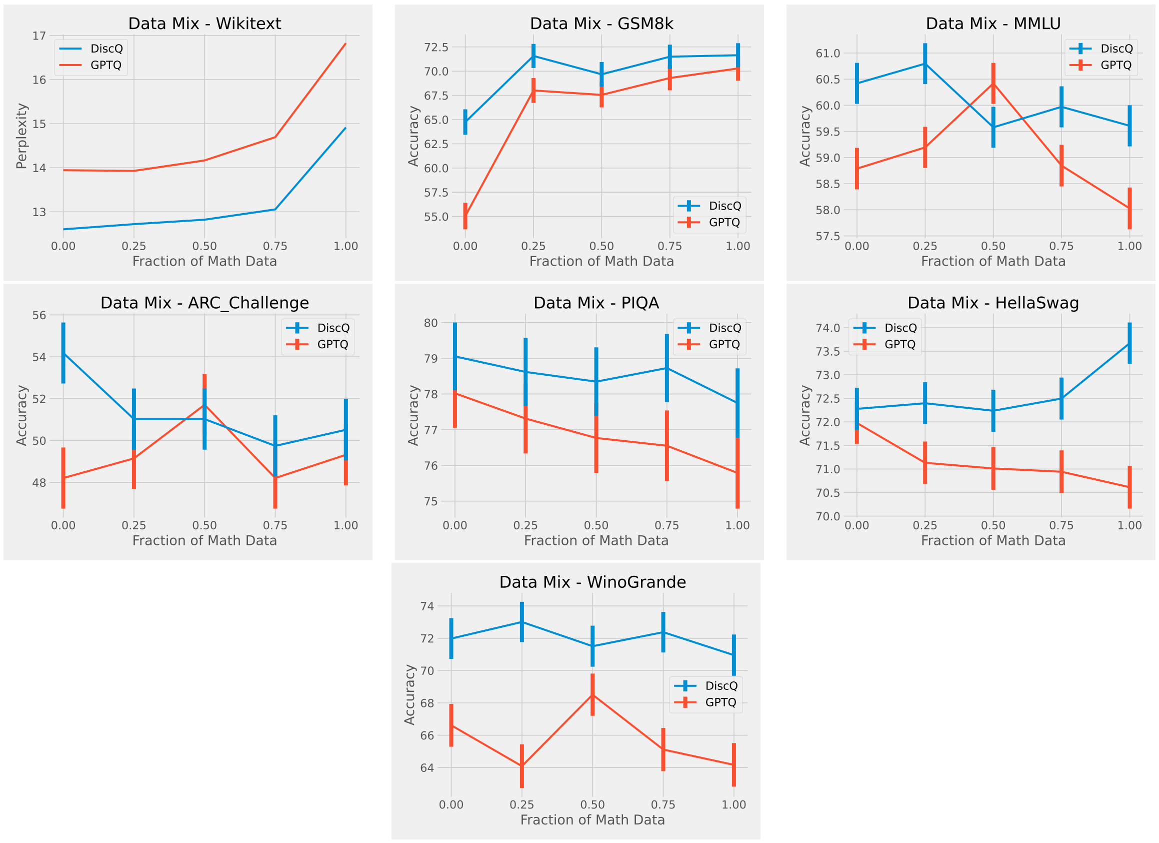

Figure 8: Effect of increasing the fraction of math data when quantizing Phi-3-mini-4k-instruct at 3.25 bits. For 8192 total samples, we use a fraction of math subject data (GSM8k and MetaMathQA), and the remaining our standard RedPajama. Across all evaluations, there is a meaningful change as a result of changing the data mix.

KL Coeff	Intermed Coeff	Intermed Type	Wiki↓	GSM8k↑
1.0	0.0	None	12.8	64.9$_{\pm 1.3}$
0.0	1.0	Layer	14.7	54.1$_{\pm 1.4}$
0.0	1.0	Linear	14.3	60.1$_{\pm 1.4}$
0.1	0.9	Linear	13.1	61.4$_{\pm 1.3}$
0.5	0.5	Linear	12.9	63.9$_{\pm 1.3}$
0.9	0.1	Linear	12.8	63.8$_{\pm 1.3}$

Table 8: Distillation Ablations. Quantizing Phi-3-mini-4k to 3.25 bits using a reduced 1024 samples of RedPajama. We test affine combinations between the KL divergence loss and intermediate L2 loss, which is either between the linear or decoder layers. Standard KL divergence does best.

E.5. Ablations

Distillation. We tried several distillation formulations, but ultimately chose a standard KL divergence between the outputs of the original and quantized model as the best approach. See Table 8.

We quantize Phi-3-mini-4k-instruct to 3.25 bits, using 1024 samples. We tune the hyper-parameters as described at the beginning of this section. Note that for these ablations we used fewer samples than in our main experiments. In addition to the standard KL divergence, we tried several intermediate loss formulations for knowledge distillation. We used a normalized L2 loss between the outputs of the teacher and student, either per decoder layer (Intermed Type = Layer), or between each linear layer (Intermed Type = Linear). This distillation formulation was presented in Kurtic et al. (2023) for recovering LLMs after pruning. We also investigated taking an affine combination between the KL and intermediate losses, trying several different coefficients. Table 8 shows our results; using just the KL divergence gives the best results. We also tried minimizing the ground truth loss instead of a distillation loss. We use the same setup as Table 8, and find that minimizing the ground truth loss achieves 52.7% GSM8k accuracy, and 13.6 Wikitext perplexity. Therefore we use the KL divergence.

Method	Wbits	Wiki↓	GSM8k↑
Base Model	16.0	9.5	$84.4_{\pm 1.0}$
rand c	3.25	27.2	$5.0_{\pm 0.6}$
$c = 1 - 2y$	3.25	12.6	$64.2_{\pm 1.3}$
rand c	3.50	23.1	$9.6_{\pm 0.8}$
$c = 1 - 2y$	3.50	12.0	$69.5_{\pm 1.3}$

Table 9: Ablation for choosing random $c \in [-1, 1]^n$ vs choosing $c = 1 - 2y$ on Phi-3-mini-4k-instruct.

λ, c hyper-parameters. We have done ablations on the choice of λ and c in the DiscQuant objective of Eq (4). On tuning λ: we found that the algorithm is not that sensitive as long as you get it right within a multiplicative factor of 2. Moreover the same choice of λ also worked across different models and bit settings of the quantized model. On tuning c: we compare choosing c at random vs $c = 1 - 2y$ as suggested in DiscQuant, see Table 9. Though theoretically random c will also work to find a vertex of the polytope, $c = 1 - 2y$ which finds the vertex closest to the original weights performs much better.