RESQ : MIXED PRECISION QUANTIZATION OF LARGE LANGUAGE MODELS WITH LOW-RANK RESIDUALS

Utkarsh Saxena[1]*, **Sayeh Sharify**[2], **Kaushik Roy**[1], **Xin Wang**[2]
[1]Purdue University, West Lafayette, USA
[2]d-Matrix, Santa Clara, USA

ABSTRACT

Quantizing weights, activations, and KV cache in large language models to 4-bit without degrading generalizability is challenging due to outlier-induced activation quantization errors. We propose *ResQ*, a PTQ method that uses principal component analysis to identify a low-rank subspace (in practice $1/8$ of the hidden dimension) and keeps coefficients within this subspace in 8-bit while quantizing the rest in 4-bit. Within each subspace, invariant random rotation is applied to further suppress outliers. ResQ outperforms recent PTQ methods on Llama and Qwen2.5, achieving up to 33% lower Wikitext perplexity than *SpinQuant* and up to $3\times$ speedup over 16-bit. Code is available at here[1].

1 INTRODUCTION

Quantization reduces storage (weight quantization), memory usage (KV cache quantization), and compute complexity (activation quantization) enabling efficient on-device inference of large language models (LLMs). While post-training methods achieve 2-bit KV cache quantization (Liu et al., 2024b; Hooper et al., 2024) and low-precision weights (Frantar et al., 2022; Lin et al., 2024c), activation quantization below 8-bit remains challenging due to large activation outliers (Dettmers et al., 2022). Recent LLM activation quantization methods feature two useful strategies: *differential treatment of outliers* retain outlier channels in high precision, leading to mixed-precision quantization (e.g., (Dettmers et al., 2022; Zhao et al., 2024; Ashkboos et al., 2024b)). Recent works, QUIK (Ashkboos et al., 2024b) and ATOM (Zhao et al., 2024), observe that activation outliers are confined to specific channels and address this by statically preserving those channels in high precision. Alternatively, another line of approaches use *invariant random rotation* to suppress outliers, leading to less difficult uniform low-precision quantization (e.g., (Ashkboos et al., 2024c; Liu et al., 2024a)). QuaRot (Ashkboos et al., 2024c) applies random Hadamard rotations, while SpinQuant (Liu et al., 2024a) optimizes rotations via gradient descent to minimize quantization error. Both types of activation quantization approaches reduce quantization error; yet a notable model performance gap persists from the 16-bit baseline.

To mend this gap, we introduce *ResQ*, a novel PTQ method that enables efficient 4-bit quantization of activations, weights, and KV cache. Specifically, by means of principal component analysis (PCA) conducted offline, we first identify a low-rank subspace that captures highest variances in activation, and mark the projected coefficients along this subspace for high-precision (8-bit) and the complement subspace for low-precision (4-bit) quantization. Then, ResQ employs invariant random rotations within each subspace before quantization to further suppress outliers. We prove this minimizes error, with most projections fused into adjacent weights for minimal overhead. Compared with related activation quantization approaches, ResQ achieves highest quantization SNR (Figure 1(b)) with its provably optimal treatment high precision component. ResQ also supports KV cache quantization and integrates with GPTQ (Frantar et al., 2022), enhancing LLM generalization. When quantizing weights, activations, and KV cache to 4-bit with only $1/8$ channels in 8-bit, ResQ reduces perplexity by 4–33% on Wikitext and improves 0-shot accuracy by 0.1–5.4% over SpinQuant (Liu et al., 2024a), without requiring gradient-based optimization. Compared with 16-bit floating point

*corresponding author : Utkarsh Saxena (saxenau@purdue.edu)
[1]https://github.com/utkarsh-dmx/project-resq

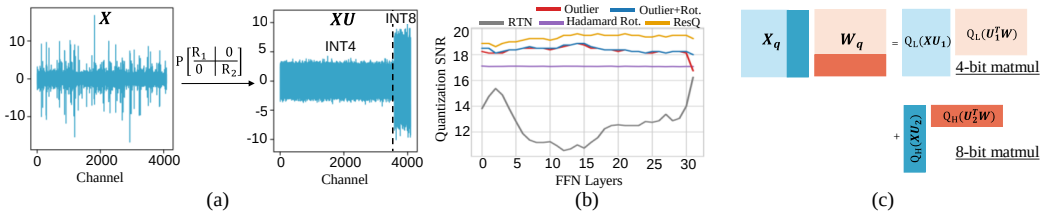

Figure 1: (a) Activation distribution before and after ResQ projections, (b) Quantization SNR for feed forward network layers (higher is better), (c) Mixed precision matrix multiplication.

model, ResQ achieves upto 3× inference speedup which is only on an average 14% less than fully `INT4` inference. Detailed description of related works is provided in Appendix A.

We claim the following contributions : (1) We propose ResQ, a mixed precision weight, activation, and KV cache quantization method by keeping low-rank, high-variance components in high precision, in combination with random rotation-induced outlier suppression. (2) We theoretically analyze the projection matrices in ResQ and show that using PCA-based projections minimizes quantization error. (3) We conduct extensive experiments on various models and language tasks and show that ResQ outperforms related state-of-the-art approaches. (4) We develop CUDA kernels and achieve runtime speedup on NVIDIA GPUs with our quantized models.

2 RESQ

In this section, we introduce ResQ, a mixed-precision quantization method that projects weights, activations, and KV cache into an orthogonal space, keeping low-rank components in 8-bit and the rest in lower precision. We detail the quantization scheme, basis generation, theoretical guarantees, and end-to-end LLM inference deployment.

2.1 QUANTIZATION SCHEME

The N-bit integer quantization and dequantization process on matrix $\boldsymbol{A}$ is given as

$$Q_N(\boldsymbol{A}) = \left\lfloor \frac{\boldsymbol{A} - z_A}{s_A} \right\rceil \cdot s_A + z_A, \tag{1}$$

where $\lfloor \cdot \rceil$ is a round-and-clip function; s_X and z_X the scale and zero-point; $z_X = 0$, $s_X = \frac{\max(|\boldsymbol{X}|)}{2^{N-1}-1}$ for symmetric quantization or $z_X = \min(\boldsymbol{X})$, $s_X = \frac{\max(\boldsymbol{X}) - \min(\boldsymbol{X})}{2^N - 1}$ for asymmetric quantization.

Given input activation $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ and weight $\boldsymbol{W} \in \mathbb{R}^{d \times d}$, we project them onto orthogonal basis $\boldsymbol{U} \in \mathbb{R}^{d \times d}$, then quantize the coefficients. High-precision components are captured by $\boldsymbol{U}_h \in \mathbb{R}^{d \times r}$, and low precision by $\boldsymbol{U}_l \in \mathbb{R}^{d \times (d-r)}$, ensuring $\boldsymbol{U}_h \boldsymbol{U}_h^\top + \boldsymbol{U}_l \boldsymbol{U}_l^\top = \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}$. The rank r controls the amount of components in high precision (in practice we typically choose $r = d/8$). The quantized activation and weights are,

$$\boldsymbol{X}_q = Q_L(\boldsymbol{X}\boldsymbol{U}_l) + Q_H(\boldsymbol{X}\boldsymbol{U}_h), \qquad \boldsymbol{W}_q = Q_L(\boldsymbol{U}_l^\top \boldsymbol{W}) + Q_H(\boldsymbol{U}_h^\top \boldsymbol{W}) \tag{2}$$

The layer output is given below and also demonstrated in Figure 1(c).

$$\boldsymbol{X}_q \boldsymbol{W}_q = Q_L(\boldsymbol{X}\boldsymbol{U}_l) Q_L(\boldsymbol{U}_l^\top \boldsymbol{W}) + Q_H(\boldsymbol{X}\boldsymbol{U}_h) Q_H(\boldsymbol{U}_h^\top \boldsymbol{W}). \tag{3}$$

Due to orthogonality, the projections preserve the original model output in absense of quantization.

2.2 PROJECTIONS AND OPTIMALITY THEREOF

The orthogonal basis $\boldsymbol{U}$ should (1) prioritize important components for high-precision quantization and (2) minimize quantization error in both high- and low-precision groups. We construct $\boldsymbol{U} = \boldsymbol{P}\boldsymbol{R}$ using two rotation matrices: $\boldsymbol{P}$ for importance based projections and $\boldsymbol{R}$ to minimize quantization error.

$$\boldsymbol{U} = \boldsymbol{P}\boldsymbol{R} = [\boldsymbol{P}_l \, \boldsymbol{P}_h] \begin{bmatrix} \boldsymbol{R}_l & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_h \end{bmatrix}, \tag{4}$$

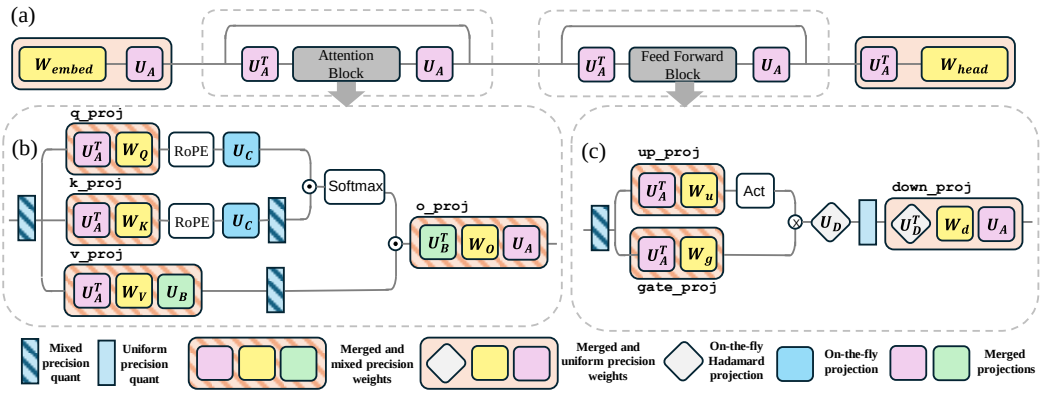

Figure 2: Model inference with ResQ incorporating the projection matrices. (a) $\boldsymbol{U}_A$ modifies the inputs across blocks enabling better quantization. (b) $\boldsymbol{U}_B, \boldsymbol{U}_C$ enables mixed precision quantization of KV cache. (c) $\boldsymbol{U}_D$ projects the activations and weights of `down_proj` layer.

Inspired by prior work (Ashkboos et al., 2024c; Chee et al., 2024), we make $\boldsymbol{R}_l, \boldsymbol{R}_h$ random orthogonal matrices because random rotation reduces outliers, making the rotated matrices easier to quantize. Furthermore, projection with a random orthogonal matrix increases Gaussianity of activations and weights within high- and low-precision groups, due to Lemma 2.1, conducive to the quantizations applied to these groups.

Lemma 2.1. *By Central Limit Theorem, the distribution after multiplication with random orthogonal matrix is approximately Gaussian (Tseng et al., 2024).*

To determine $\boldsymbol{P}$, we minimize the activation quantization error $\|\boldsymbol{X} - \boldsymbol{X}_q\|_F$. For activations quantized according to Equation 2, we have,

$$\|\boldsymbol{X} - \boldsymbol{X}_q\|_F = \|\boldsymbol{X}\boldsymbol{U}_l - Q_L(\boldsymbol{X}\boldsymbol{U}_l)\|_F + \|\boldsymbol{X}\boldsymbol{U}_h - Q_H(\boldsymbol{X}\boldsymbol{U}_h)\|_F. \tag{5}$$

Theorem 2.2. *For any matrix $\boldsymbol{X}$ quantized to $\boldsymbol{X}_q$ according to method described in Equation 2, assuming the values to be quantized in $\boldsymbol{X}$ are normally distributed, we have*

$$\mathbb{E}\|\boldsymbol{X} - \boldsymbol{X}_q\|_F \leq \frac{\sqrt{\pi \log(d-r)}}{2^{L-1}-1}\mathbb{E}\|\boldsymbol{X}\|_F - \left[\frac{\sqrt{\pi \log(d-r)}}{2^{L-1}-1} - \frac{\sqrt{\pi \log r}}{2^{H-1}-1}\right] \mathbb{E}\|\boldsymbol{X}\boldsymbol{P}_h\|_F. \tag{6}$$

Full proof of Theorem 2.2 is in Appendix B. Theorem 2.2 bounds the quantization error, which can be minimized by maximizing $\|\boldsymbol{X}\boldsymbol{P}_h\|_F$. This occurs when $\boldsymbol{P}_h$ consists of top eigenvectors of covariance matrix of activations $\boldsymbol{X}\boldsymbol{X}^\top$. Thus, the low rank space for high precision quantization can be obtained by means of PCA. To facilitate that, we obtained $\boldsymbol{X}\boldsymbol{X}^\top$ using a calibration dataset and perform PCA to obtain $\boldsymbol{P}_h$. Note that this is done offline and once obtained, $\boldsymbol{P}_h$ does not change. The subspace for low-precision quantization can be obtained using $\boldsymbol{U}_h\boldsymbol{U}_h^\top + \boldsymbol{U}_l\boldsymbol{U}_l^\top = \boldsymbol{P}_h\boldsymbol{P}_h^\top + \boldsymbol{P}_l\boldsymbol{P}_l^\top = \boldsymbol{I}$ (because $\boldsymbol{R}_i$ is orthogonal). If we construct $\boldsymbol{P}$ by taking eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$ arranged in *increasing* order of eigenvalues, the last r columns of such a $\boldsymbol{P}$ would correspond to $\boldsymbol{P}_h$ and the first $d-r$ columns would correspond to $\boldsymbol{P}_l$. The distribution of activation after before and after projection is given in Figure 1(a).

2.3 Inference computation with optimized projections

Once projection matrices are obtained, activations and weights are projected using $\boldsymbol{U}$. Weights are projected and quantized offline, while activation projections are merged into the previous layer's weights for efficiency. Decoder-based LLMs require four projections (Figure 2): $\boldsymbol{U}_A$ (hidden dim), $\boldsymbol{U}_B, \boldsymbol{U}_C$ (attention head dim), and $\boldsymbol{U}_D$ (FFN hidden dim).

Block Boundary Projections Activations for attention and FFN are projected via $\boldsymbol{U}_A$ by right-multiplying the final linear layer weights (`o_proj` in attention, `down_proj` in FFN),

incurring no extra inference cost. To ensure numerical invariance, the first linear layers (`q_proj|k_proj|v_proj` in attention, `up_proj|gate_proj` in FFN) are pre-multiplied with $\boldsymbol{U}_A^\top$. Embedding and final head weights are also adjusted for residual stream projection.

Attention Block Projections $\boldsymbol{U}_B, \boldsymbol{U}_C$ project activations within the attention block (Figure 2b). Post-multiplying the value projection layer by $\boldsymbol{U}_B$ ensures optimal KV cache quantization, requiring `o_proj` weights to be pre-multiplied by $\boldsymbol{U}_B^\top$ for numerical invariance. $\boldsymbol{U}_C$ optimally quantizes keys by projecting both query and key, preserving the attention dot product:

$$\boldsymbol{q}_{\text{proj}}\boldsymbol{K}_{\text{proj}}^\top = (\boldsymbol{q}\boldsymbol{U}_C)(\boldsymbol{U}_C^\top\boldsymbol{K}^\top) = \boldsymbol{q}\boldsymbol{K}^\top. \tag{7}$$

Since $\boldsymbol{U}_C$ cannot be merged due to RoPE, the projection is explicitly computed at runtime, but made more efficient by applying uniform precision quantization to $\boldsymbol{U}_C$ and corresponding inputs.

FFN Block Projections $\boldsymbol{U}_D$ enhances FFN activation quantization (Figure 2c). $\boldsymbol{U}_D^\top$ is left-multiplied with `down_proj` weights, but due to activation functions, $\boldsymbol{U}_D$ cannot be merged with preceding layers and is also computed at runtime. Since FFN hidden dimensions (d_{FFN}) are 3–4× the embedding size, direct multiplication with $\boldsymbol{U}_D$ is costly. To reduce overhead, $\boldsymbol{U}_D$ is set as a Hadamard matrix for efficient transforms, while `down_proj` weights and activations are uniformly quantized to low precision.

3 Experiments

3.1 Setup

Models, Tasks, Datasets, and Baselines We evaluate ResQ on Llama 3 (Meta, 2024b), Llama 3.2 (Meta, 2024a), Qwen2.5 (Yang et al., 2024a), and multi-modal Qwen2 VL models (Wang et al., 2024). Baselines include GPTQ (Frantar et al., 2022), QuaRot (Ashkboos et al., 2024c), QUIK (Ashkboos et al., 2024b), SpinQuant (Liu et al., 2024a), and SmoothQuant+, a stronger baseline created by combining SmoothQuant (Xiao et al., 2023) with GPTQ following Sharify et al. (2024). We assess quantization on *language modeling* (Wikitext (Merity et al., 2016)), *reasoning* (average 0-shot accuracy on Arc-c/e (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), Openbook QA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021)), *understanding* (MMLU (Hendrycks et al., 2021)), *math* (GSM8K (Cobbe et al., 2021)), *summarization* (samsum, qmsum from LongBench (Bai et al., 2024)), *code completion* (repobench-p (Liu et al., 2023b)), and *multi-modal* (MMMU (Yue et al., 2024)).

Implementation details We implement ResQ using the HuggingFace Transformers library (Wolf et al., 2020) with PyTorch (Paszke et al., 2019). We share a single $\boldsymbol{U}_A$ across all layers, while $\boldsymbol{U}_B$, $\boldsymbol{U}_C$ and $\boldsymbol{U}_D$ are generated per layer. Following SpinQuant (Liu et al., 2024a), we use per-token asymmetric activation quantization, per-channel symmetric weight quantization, and per-head asymmetric KV cache quantization. Projection matrices $\boldsymbol{U}_A, \boldsymbol{U}_B, \boldsymbol{U}_D$ are fused into weights, with GPTQ (Frantar et al., 2022) applied for weight quantization.To enable efficient on-the-fly projections, $\boldsymbol{U}_D$ is a Hadamard matrix, and $\boldsymbol{U}_C$ and its activations are quantized to 8-bit. The full process runs on a single NVIDIA A100, taking 35 minutes for `Meta-Llama-3-8B`. See Appendix C for details.

3.2 Results

Language modeling, understanding, and reasoning tasks The results are presented in Table 1. We see that ResQ reduces the gap to 16-bit performance and outperforms the quantization baselines across all tasks on all models. Particularly, on Llama 3/3.2 family of models, ResQ outperforms SpinQuant by achieving 4-33% lower Wikitext perplexity, 0.1-5.4% better average 0-shot accuracy and a 1-14.5% better accuracy on MMLU benchmark without any additional training. For the Qwen-2.5 model family, all other baselines fail to achieve competitive results, and ResQ significantly outperforms them. Compared with QUIK, another mixed precision quantization approach, ResQ achieves 42-50% better Wikitext perplexity, 5.8-12.3% better average zero shot accuracy and 4.3-24.5% better MMLU accuracy over all models. Full results in App. D.

Table 1: Perplexity on Wikitext, average 0-shot common sense reasoning accuracy, and average 0-shot MMLU accuracy at W/A/KV = 4-bit. *: Mixed precision with 1/8 channels in 8-bit for W/A/KV = 4.5-bit. ↑: higher is better, ↓: lower is better. Full results in Appendix D.

Method	Meta-Llama-3-8B Wiki (↓)	Meta-Llama-3-8B Reasoning (↑)	Meta-Llama-3-8B MMLU (↑)	Meta-Llama-3-70B Wiki (↓)	Meta-Llama-3-70B Reasoning (↑)	Meta-Llama-3-70B MMLU (↑)
Llama 3						
16-bit	6.1	67.1	63.1	2.9	73.1	75.9
RTN	218.9	39.3	23.6	452.7	45.5	23.2
GPTQ	166.3	39.8	23.3	1.2e4	34.9	25.5
SQ+	78.2	42.5	24.7	-	-	-
QUIK*	14.2	51.6	32.7	8.0	58.2	51.1
QuaRot	7.8	62.1	53.2	5.7	67.6	65.3
SpinQuant	7.4	63.8	56.2	6.2	65.7	59.4
ResQ*	**7.1**	**63.9**	**57.2**	**4.1**	**71.1**	**73.9**
Llama 3.2						
Method	**Llama-3.2-1B Wiki (↓)**	**Llama-3.2-1B Reasoning (↑)**	**Llama-3.2-1B MMLU (↑)**	**Llama-3.2-3B Wiki (↓)**	**Llama-3.2-3B Reasoning (↑)**	**Llama-3.2-3B MMLU (↑)**
16-bit	9.8	54.9	36.9	7.8	62.7	54.8
RTN	329.1	38.1	23.8	268.8	38.7	25.7
GPTQ	108.9	38.0	24.9	178.3	40.3	24.8
SQ+	228.9	38.0	24.1	96.1	39.0	25.9
QUIK*	21.8	44.3	25.1	15.8	48.8	31.1
QuaRot	14.3	49.0	25.5	10.1	56.1	42.0
SpinQuant	13.6	48.8	25.6	9.2	57.9	44.2
ResQ*	**12.4**	**50.1**	**29.4**	**8.8**	**59.0**	**49.8**
Qwen2.5						
Method	**Qwen2.5-3B Wiki (↓)**	**Qwen2.5-3B Reasoning (↑)**	**Qwen2.5-3B MMLU (↑)**	**Qwen2.5-72B Wiki (↓)**	**Qwen2.5-72B Reasoning (↑)**	**Qwen2.5-72B MMLU (↑)**
16-bit	8.0	63.8	66.1	3.9	73.4	84.3
RTN	3.9e4	35.1	23.4	4.5e4	34.3	24.0
GPTQ	9.9e3	35.1	23.2	3.8e4	34.5	23.3
SQ+	7.3e4	34.8	23.9	-	-	-
QUIK*	15.5	51.2	39.4	8.3	61.9	69.3
QuaRot	68.8	47.7	28.9	4.9	70.3	80.1
ResQ*	**9.0**	**61.1**	**61.2**	**4.6**	**72.0**	**81.5**

Table 2: Comparison of performance of quantization approaches on generative tasks at precisions of W/A/KV = 4-bit. *: Mixed precision with 1/8 channels in 8-bit for W/A/KV = 4.5-bit. ↑: higher is better.

Model	Method	GSM8K 5-shot (↑) flexible-e	GSM8K 5-shot (↑) strict-m	LongBench (↑) qmsum	LongBench (↑) samsum	LongBench (↑) repo-p
Meta-Llama-3-8B	16-bit	51.0	50.6	23.9	44.8	66.4
	QUIK*	2.3	0.0	10.5	25.2	37.6
	QuaRot	27.6	27.1	22.0	43.8	60.6
	SpinQuant	29.8	29.6	23.0	43.9	**62.6**
	ResQ*	**33.6**	**33.2**	**23.1**	**44.1**	62.3
Llama-3.2-8B	16-bit	25.1	24.9	23.1	43.0	64.4
	QUIK*	2.5	0.0	15.9	31.7	30.9
	QuaRot	10.1	9.1	20.6	39.5	56.8
	SpinQuant	11.6	11.4	**21.7**	41.9	59.1
	ResQ*	**17.1**	**16.7**	**21.7**	**43.0**	**61.5**

Table 3: 0-shot MMMU accuracy (higher is better) of vision language models when quantized using various approaches. *: Mixed precision with 1/8 channels in 8-bit and rest in 4-bit.

W/A/KV	Method	Model: Qwen2-VL-2B-Instruct	Model: Qwen2-VL-7B-Instruct
16/16/16	Baseline	39.6	51.6
4/4/4	RTN	25.0	26.7
	GPTQ	27.7	24.9
	QuaRot	24.0	24.5
	QUIK*	26.3	28.9
4.5/4.5/4.5	ResQ*	**29.7**	**47.0**
4/8/4	RTN	24.9	25.2
	GPTQ	23.4	24.3
	QuaRot	26.5	24.5
	QUIK*	28.4	26.4
4.5/8/4.5	ResQ*	**34.0**	**48.8**

Generative tasks We evaluate ResQ on auto-regressive tasks, including GSM8K (math), dialogue summarization (qmsum, samsum), and code completion (repobench-p, Table 2), to assess generation across domains. On the challenging GSM8K benchmark where QUIK fails to produce meaningful results, ResQ outperforms SpinQuant by 3.8% and 5.5% on the 8B and 3B parameter model respectively, closing the gap to the 16-bit baseline. On LongBench evaluation tasks, ResQ demonstrates competitive performance and outperforms SpinQuant without any additional training.

Multi-modal understanding We benchmark the quantization approaches on vision language models (VLMs) by quantizing Qwen2 VL family and evaluating their performance on MMMU (Table 3, Yue et al. 2024). Only the language model is quantized while the vision encoder remains in 16-bit as the language model has many more parameters (over 10× for `Qwen2-VL-7B-Instruct`). ResQ outperforms baselines on both 2B and 7B models, achieving superior accuracy and demonstrating its generalizability. Results for individual MMMU tasks are provided in Appendix E.

Table 4: Perplexity on Wikitext, average 0-shot common sense reasoning accuracy, and average 0-shot MMLU accuracy at iso-bitwidth W/A/KV = 4-bit. ↑: higher is better, ↓: lower is better. Full results in Appendix F.

Method	Qwen2.5-1.5B Wiki (↓)	Qwen2.5-1.5B Reasoning (↑)	Qwen2.5-1.5B MMLU (↑)	Qwen2.5-3B Wiki (↓)	Qwen2.5-3B Reasoning (↑)	Qwen2.5-3B MMLU (↑)
QuaRot	6599.9	38.3	23.5	68.8	47.7	28.9
SpinQuant	5938.3	38.1	23.7	70.6	48.6	32.8
ResQ	**187.3**	**46.6**	**27.2**	**9.8**	**59.1**	**52.2**
Method	**Qwen2.5-7B Wiki (↓)**	**Qwen2.5-7B Reasoning (↑)**	**Qwen2.5-7B MMLU (↑)**	**Qwen2.5-14B Wiki (↓)**	**Qwen2.5-14B Reasoning (↑)**	**Qwen2.5-14B MMLU (↑)**
QuaRot	4035.9	38.4	24.1	6.8	67.1	70.9
SpinQuant	3395.4	38.6	24.3	6.6	67.4	70.1
ResQ	**34.2**	**56.2**	**58.0**	**6.5**	**67.5**	**71.3**
Method	**Qwen2.5-32B Wiki (↓)**	**Qwen2.5-32B Reasoning (↑)**	**Qwen2.5-32B MMLU (↑)**	**Qwen2.5-72B Wiki (↓)**	**Qwen2.5-72B Reasoning (↑)**	**Qwen2.5-72B MMLU (↑)**
QuaRot	6.1	67.8	77.0	**4.9**	70.3	**80.1**
SpinQuant	6.0	67.9	77.6	-	-	-
ResQ	**5.9**	**69.1**	**77.9**	**4.9**	**71.1**	**80.1**

Iso-bitwidth comparison We also compare ResQ with QuaRot (Ashkboos et al., 2024c) and SpinQuant (Liu et al., 2024a) under iso-bitwidth settings of W/A/KV = 4-bit. To enable average 4-bit in ResQ, we refer to construction of projection matrix $\boldsymbol{P}$ which is formed by obtaining eigen vectors of activations in *increasing* order of eigen values (sec. 2.2). We keep first 1/8 channels corresponding to eigen vectors with

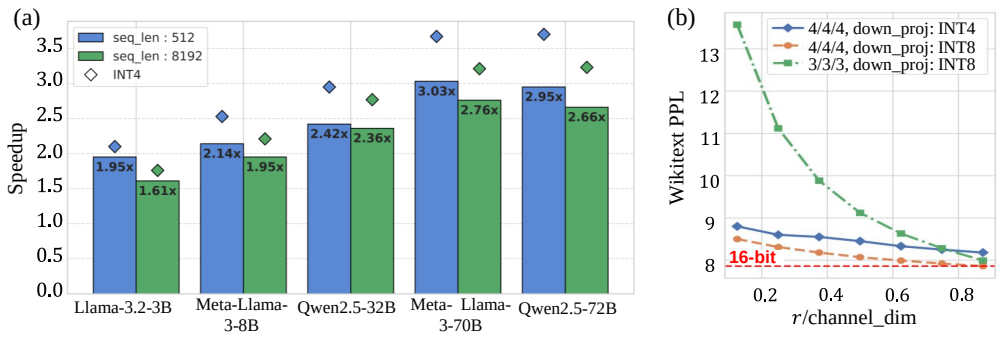

Figure 3: (a) Per block prefill speedup of ResQ and INT4 kernel on NVIDIA RTX 3090 over 16-bit floating point baseline for batch size of 1. (b) Ablation study on changing rank of high precision subspace for `Llama-3.2-3B`

lowest eigen value of $\boldsymbol{P}$ in 2-bit, last $1/8$ channels corresponding to top eigen vectors of $\boldsymbol{P}$ in 6-bit, and the remaining channels in 4-bit. Within each quantization group (2-bit, 4-bit, 6-bit) we apply random rotation to minimize quantization error. Inference with these modified projection matrices remains unchanged. Table 4 shows comparison with QuaRot and SpinQuant for Qwen-2.5 family of models. For each model size, ResQ outperforms both QuaRot and SpinQuant on variety of language modeling, understanding and reasoning benchmarks. Specifically, on challenging MMLU benchmark ResQ achieves upto 34% absolute improvement in performance. Task wise results on reasoning and MMLU benchmarks are provided in Appendix F.

Hardware Performance We implement the mixed-precision quantization using CUDA 11.8 and PyTorch. We use `CUTLASS` (Thakkar et al., 2023) to perform `INT4` and `INT8` GEMM operations on TensorCore. On NVIDIA RTX 3090 GPU, representative of resource constrained setting, we achieve about $1.61\times$ to $3.03\times$ speedup with ResQ over 16-bit baseline for a single decoder block tested across several language models (Figure 3). We observe higher speedup on larger models and smaller sequence lengths. Compared with INT4 implementation, ResQ kernel is only 14% slower on average across models and sequence lengths demonstrating minimal overhead associated with mixed precision computations and on-the-fly projections introduced in ResQ.

Rank of high-precision subspace ResQ allows for seemless trade-off between accuracy and performance by modulating the rank r of high precision subspace (Figure 3(b)). Increasing the rank improves perplexity albeit at the cost of increased computations in high precision.

4 Conclusion

We introduce *ResQ*, a novel mixed-precision, accelerator-friendly PTQ technique toward 4-bit quantization of large language models. ResQ projects weight, activation, and KV cache tensors to subspaces spanned by principal components, quantizing a low-rank ($1/8$ of hidden dimension) high-variance subspace to 8-bit and the rest to 4-bit. ResQ outperforms both uniform- and mixed-precision quantization methods. We demonstrate the effectiveness of ResQ across a variety of tasks—including language modeling, language understanding, common-sense reasoning, language generation and multi modal understanding—using the Llama and Qwen models. Compared to SpinQuant, the strongest baseline, ResQ achieves up to 33% lower perplexity on the WikiText dataset without requiring any additional training and offers up to $3.03\times$ speedup over the 16-bit baseline.

Acknowledgements

The authors would like to thank Wanzin Yazar and Tristan Webb for infrastructure and technical assistance and Zifei Xu and Sakshi Choudhary for helpful discussions. The authors also thank Amogh Joshi for providing access to personal NVIDIA RTX 3090. This work was supported by the Center for the Co-Design of Cognitive Systems (COCOSYS), a DARPA sponsored JUMP center of Semiconductor Research Corporation (SRC), Intel, SRC AIHW Program.

REFERENCES

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. *arXiv:2401.15024*, 2024a.

Saleh Ashkboos, Ilia Markov, Elias Frantar, Tingxuan Zhong, Xincheng Wang, Jie Ren, Torsten Hoefler, and Dan Alistarh. QUIK: Towards end-to-end 4-bit inference on generative large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3355–3371. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.197. URL https://aclanthology.org/2024.emnlp-main.197/.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240. Curran Associates, Inc., 2024c. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b5b939436789f76f08b9d0da5e81af7c-Paper-Conference.pdf.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection. *arXiv:2407.21118*, 2024.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. QuIP: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36, 2024.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized clipping activation for quantized neural networks. *arXiv:1805.06085*, 2018.

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

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35: 30318–30332, 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. *arXiv:2306.03078*, 2023.

Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. QAQ: Quality adaptive quantization for llm kv cache. *arXiv:2403.04643*, 2024.

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

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

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, 07 2024. URL https://zenodo.org/records/12608602.

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

Ziyi Guan, Hantao Huang, Yupeng Su, Hong Huang, Ngai Wong, and Hao Yu. APTQ: Attention-aware post-training mixed-precision quantization for large language models. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pp. 1–6, 2024.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. ZipCache: Accurate and efficient kv cache quantization with salient token identification. *arXiv:2405.14256*, 2024.

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

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length llm inference with kv cache quantization. *arXiv:2401.18079*, 2024.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Xianglong Liu, Luca Benini, Michele Magno, and Xiaojuan Qi. SliM-LLM: Salience-driven mixed-precision quantization for large language models. *arXiv:2405.14917*, 2024.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pp. 4466–4475, 2021.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm. *arXiv:2403.05527*, 2024.

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

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

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv:2411.05007*, 2024. URL https://arxiv.org/abs/2411.05007.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV: Adaptive kv compression via trainable orthogonal projection. *arXiv:2410.14731*, 2024a.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

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

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv:2405.04532*, 2024d.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. QLLM: Accurate and efficient low-bitwidth quantization for large language models. *arXiv:2310.08041*, 2023a.

Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking repository-level code auto-completion systems. *arXiv:2306.03091*, 2023b. URL https://arxiv.org/abs/2306.03091.

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: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. *arXiv:2402.02750*, 2024b.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv:1609.07843*, 2016.

Meta. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, 2024a. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Meta. Introducing Meta Llama 3: The most capable openly available LLM to date., 2024b. URL https://ai.meta.com/blog/meta-llama-3/.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Gunho Park, Baeseong Park, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. nuQmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv:2206.09557*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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

Charbel Sakr and Brucek Khailany. ESPACE: Dimensionality reduction of activations for model compression. *arXiv:2410.05437*, 2024.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for KV cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-emnlp.899. URL https://aclanthology.org/2024.findings-emnlp.899.

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. *arXiv:2308.13137*, 2023.

Sayeh Sharify, Utkarsh Saxena, Zifei Xu, Wanzin Yazar, Ilya Soloveychik, and Xin Wang. Post training quantization of large language models with microscaling formats. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*, pp. 241–258. PMLR, 2024.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. CUTLASS, January 2023. URL https://github.com/NVIDIA/cutlass.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv:2402.04396*, 2024. URL https://arxiv.org/abs/2402.04396.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv:2409.12191*, 2024. URL https://arxiv.org/abs/2409.12191.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv:1910.03771*, 2020.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168, 2023.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv:2412.15115*, 2024a.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No Token Left Behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv:2402.18096*, 2024b.

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

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

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv:2312.05821*, 2023b.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv:2311.16502*, 2024. URL https://arxiv.org/abs/2311.16502.

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

Chao Zeng, Songwei Liu, Yusheng Xie, Hong Liu, Xiaojian Wang, Miao Wei, Shu Yang, Fangmin Chen, and Xing Mei. ABQ-LLM: Arbitrary-bit quantized inference acceleration for large language models. *arXiv:2408.08554*, 2024.

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

A Related Works

A.1 Quantization of LLMs

Quantization reduces model size and accelerates inference by lowering neural network bit precision (Choi et al., 2018; Hubara et al., 2021; Yao et al., 2022; Park et al., 2022; Gholami et al., 2022; Xi et al., 2023). It is broadly categorized into two categories: *uniform precision quantization* (UPQ) and *mixed precision quantization* (MPQ). **Uniform precision quantization (UPQ)** applies the same bit-width across all layers, simplifying implementation but neglecting layer-specific sensitivity to quantization. **Weight-only UPQ** methods reduce storage by compressing weights, using techniques like Hessian-guided rounding (GPTQ, Frantar et al. 2022), adaptive rounding (QuIP, Chee et al. 2024), channel-wise scaling (AWQ, Lin et al. 2024c), and multi-codebook quantization (AQLM, Egiazarian et al. 2024). However, these methods struggle with batch processing due to significant activation memory overhead. **Weight-activation UPQ** compresses both weights and activations to address this. Methods such as SmoothQuant (Xiao et al., 2023) and OmniQuant (Shao et al., 2023) scale activations and weights to handle outliers, while RPTQ (Yuan et al., 2023a), QLLM (Liu et al., 2023a), and QServe (Lin et al., 2024d) employ channel-level strategies like clustering and reordering. Rotation-based methods such as QuaRot (Ashkboos et al., 2024c), SpinQuant (Liu et al., 2024a) and DuQuant (Lin et al., 2024b) further enhance robustness in low-precision scenarios. **KV cache UPQ** reduces memory for large batches or long contexts. FlexGen (Sheng et al., 2023) employs 4-bit quantization and memory offloading, while KIVI (Liu et al., 2024b) uses asymmetric 2-bit quantization for compression, enabling efficient inference.

Mixed precision quantization (MPQ) optimizes bit-widths by adapting to the sensitivity of weights and activations, achieving better accuracy than UPQ at similar compression rates. *Our proposed method, ResQ, follows the MPQ approach.* **Weight-only MPQ** has advanced efficiency for memory-bound applications with minimal activation demands. Methods like OWQ (Lee et al., 2024) and SpQR (Dettmers et al., 2023) mitigate activation outliers' impact by retaining critical features in full precision, while SqueezeLLM (Kim et al., 2023) employs Dense-and-Sparse decomposition to efficiently store sensitive weights. **Weight-activation MPQ** enhances efficiency by addressing activation outliers (e.g. Guan et al. 2024; Zeng et al. 2024). Methods like LLM.int8() (Dettmers et al., 2022) and QUIK(Ashkboos et al., 2024b) preserve critical activations with mixed or low-precision decompositions, while Atom (Zhao et al., 2024) and SliM-LLM (Huang et al., 2024) optimize quantization through channel reordering and salience-driven bit allocation. **KV cache MPQ** reduces memory usage while preserving precision for critical tokens using techniques like non-uniform quantization, importance-aware precision, and salient token compression (Hooper et al., 2024; Yang et al., 2024b; Dong et al., 2024; He et al., 2024). Alternatively, GEAR quantizes all tokens' KV cache and maintains low-rank quantization error (Kang et al., 2024).

A.2 LOW-RANK DECOMPOSITION

Low-rank decomposition techniques have been widely used in model compression, reducing dimensionality while maintaining performance. For instance, SliceGPT (Ashkboos et al., 2024a) projects weight matrices onto principal components for sparsification, while ESPACE (Sakr & Khailany, 2024) reduces activation dimensionality via pre-calibrated projections, achieving inference-time efficiency. Similarly, ASVD (Yuan et al., 2023b) introduces an activation-aware decomposition method that incorporates activation distributions into weight decomposition. Additionally, low-rank decomposition can be applied to reduce KV cache size. For example, Eigen Attention (Saxena et al., 2024) and ASVD (Yuan et al., 2023b) employ low-rank approximations to reduce memory usage in KV caches during attention operations. PALU (Chang et al., 2024) introduces learnable projections to adaptively compress KV caches based on the compression budget. Finally, Matryoshka KV Cache (Lin et al., 2024a) refines this with hierarchical orthogonal projections and knowledge distillation.

B PROOF OF THEOREM 2.2

We begin the proof by introducing the following lemma.

Lemma B.1. *For any tensor $\boldsymbol{R}$ quantized following the quantization described in equation 1, assuming the values of $\boldsymbol{R}$ follows a normal distribution, we have*

$$\mathbb{E}\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F \leq \frac{\sqrt{\pi \log\left[size(\boldsymbol{R})\right]}}{2^{n-1}-1}\mathbb{E}\|\boldsymbol{R}\|_F \tag{8}$$

where size($\boldsymbol{R}$) denotes the number of elements in $\boldsymbol{R}$.

Proof of lemma B.1 can be found in Li et al. (2024). From this lemma we obtain that the quantization error $\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F$ is bounded by the magnitude of the tensor quantized $\|\boldsymbol{R}\|_F$. Now for our use case of mixed precision quantization where the low-precision component is quantized to L bits and high precision component is quantized to H bits, we write the quantization error again below,

$$\begin{aligned}\mathbb{E}\|\boldsymbol{X} - \boldsymbol{X}_q\|_F = \mathbb{E}\|\boldsymbol{X}\boldsymbol{U}_l - Q_L(\boldsymbol{X}\boldsymbol{U}_l)\|_F \\ + \mathbb{E}\|\boldsymbol{X}\boldsymbol{U}_h - Q_H(\boldsymbol{X}\boldsymbol{U}_h)\|_F.\end{aligned} \tag{9}$$

The random rotation matrices $\boldsymbol{R}$ ensure that $\boldsymbol{X}\boldsymbol{U}_l$ and $\boldsymbol{X}\boldsymbol{U}_h$ are normally distributed by Lemma 2.1. Applying Lemma B.1 to the quantization error in equation 9, we get,

$$\begin{aligned}\|\boldsymbol{X} - \boldsymbol{X}_q\|_F &\leq \frac{\sqrt{\log(\text{size}(\boldsymbol{X}\boldsymbol{U}_l))\pi}}{2^{L-1}-1}\mathbb{E}||\boldsymbol{X}\boldsymbol{U}_l||_F \\ &+ \frac{\sqrt{\log(\text{size}(\boldsymbol{X}\boldsymbol{U}_h))\pi}}{2^{H-1}-1}\mathbb{E}||\boldsymbol{X}\boldsymbol{U}_h||_F \\ &= \frac{\sqrt{\log(\text{size}(\boldsymbol{X}\boldsymbol{P}_l))\pi}}{2^{L-1}-1}\mathbb{E}||\boldsymbol{X}\boldsymbol{P}_l||_F \\ &+ \frac{\sqrt{\log(\text{size}(\boldsymbol{X}\boldsymbol{P}_h))\pi}}{2^{H-1}-1}\mathbb{E}||\boldsymbol{X}\boldsymbol{P}_h||_F \\ &= \frac{\sqrt{\log(\text{size}(\boldsymbol{X}\boldsymbol{P}_l))\pi}}{2^{L-1}-1}\mathbb{E}||\text{tr}(\boldsymbol{X}\boldsymbol{P}_l\boldsymbol{P}_l^\top\boldsymbol{X}^\top)||_F \\ &+ \frac{\sqrt{\log(\text{size}(\boldsymbol{X}\boldsymbol{P}_h)\pi}}{2^{H-1}-1}\mathbb{E}||\text{tr}(\boldsymbol{X}\boldsymbol{P}_h\boldsymbol{P}_h^\top\boldsymbol{X}^\top)||_F\end{aligned} \tag{10}$$

We know size$(\boldsymbol{X}\boldsymbol{P}_l) = d - r$ and size$(\boldsymbol{X}\boldsymbol{P}_h) = r$ since r components are in high precision. With $\boldsymbol{P}_l\boldsymbol{P}_L^\top + \boldsymbol{P}_h\boldsymbol{P}_h^\top = \boldsymbol{I}$, we have

$$\begin{aligned}\|\boldsymbol{X} - \boldsymbol{X}_q\|_F &\leq \frac{\sqrt{\log(\text{d-r})\pi}}{2^{L-1}-1}(\mathbb{E}\|\boldsymbol{X}\|_F - \mathbb{E}\|\boldsymbol{X}\boldsymbol{P}_h\|_F) \\ &+ \frac{\sqrt{\log(\text{r})\pi}}{2^{H-1}-1}\mathbb{E}\|\boldsymbol{X}\boldsymbol{P}_h\|_F \\ &= \frac{\sqrt{\log(\text{d-r})\pi}}{2^{L-1}-1}\mathbb{E}\|\boldsymbol{X}\|_F \\ &- (\frac{\sqrt{\log(\text{d-r})\pi}}{2^{L-1}-1} - \frac{\sqrt{\log(\text{r})\pi}}{2^{H-1}-1})\mathbb{E}\|\boldsymbol{X}\boldsymbol{P}_h\|_F\end{aligned} \tag{11}$$

Table 5: Time taken (in NVIDIA A100 GPU hours) to quantize the model. All approaches use GPTQ for weight quantization. SpinQuant uses 4 GPUs to optimize rotation matrices.

	QuaRot	ResQ	SpinQuant
`Llama-3.2-1B`	4m	7m	13m
`Llama-3.2-3B`	8m	16m	38m
`Meta-Llama-3-8B`	17m	35m	1h41m
`Llama-2-7b-hf`	15m	33m	1h37m
`Llama-2-13b-hf`	23m	1h	3h42m

Since $\frac{\sqrt{\log(\text{d-r})\pi}}{2^{L-1}-1} - \frac{\sqrt{\log(\text{r})\pi}}{2^{H-1}-1} > 0$ the quantization error is reduced by maximizing $\|\boldsymbol{X}\boldsymbol{P}_h\|_F$

C Additional implementation details

In this work, obtaining the projection matrices and quantization of weights for **all the models** is performed on a single NVIDIA A100 80GB GPUs. Time taken by ResQ compared with other approaches is shown in Table 5. Evaluation on various benchmarks for all the models is also done on a single NVIDIA A100 GPU with the sole exception of `Meta-Llama-3-70b` which requires 4 GPUs for evaluation. We use `lm_evaluation_harness` version 0.4.5 (Gao et al., 2024) and LongBench (Bai et al., 2024) for all the evaluation tasks. For Arc-c/e, Hellaswag, OpenBook QA, PIQA tasks we report `acc_norm` while for BoolQ, SIQA and Winogrande we report `acc`.

For calibration data, we use 512 randomly choses samples for Wikitext to obtain the projection matrices. While for GPTQ we use 128 randomly choses samples from Wiktiext following the original work Frantar et al. (2022).

The KV cache, as well as the weights and activations of all Linear layers (except `mlp.down_proj`), are quantized to 4-bit precision, with $\frac{1}{8}$ of channels retained in 8-bit precision. While, the weights and activations within `down_proj` are uniformly quantized to 4-bit precision. Following Ashkboos et al. (2024c) and Liu et al. (2024a), we keep query vector in 16-bit.

D Complete results of main result tables

Detailed results of Table 1 in the main paper, including more models and task-by-task performance, are shown in Tables 6 (Llama families) and 7 (Qwen2.5 family). As expected, ResQ achieves superior performance to baselines across the series of common sense reasoning and MMLU tasks.

E Complete results of the MMMU benchmark

This section presents task-by-task results for the MMMU benchmark across six subjects—Art & Design, Business, Science, Health & Medicine, Humanities & Social Science, and Tech & Engineering—for the Qwen2 VL family when quantized to W/A/KV = 4/4/4 bits and W/A/KV = 4/8/4 bits of precision. ResQ and QUIK (Ashkboos et al., 2024b) keep $1/8$ channels in 8-bit for average 4.5-bits. On average, ResQ consistently outperforms all baselines across different models. Notably, the advantage of ResQ becomes more pronounced with larger models. For instance, for `Qwen2-VL-7B-Instruct` at W/A/KV = 4.5/8/4.5 bits of precision, ResQ achieves an average accuracy score of 48.8, significantly outperforming the next-best method, QUIK, which scores 26.4, representing an $\sim 85\%$ relative improvement.

F Complete results of iso-bitwidth comparison

Detailed results of Table 4 in the main paper are shown in Tables 9. ResQ achieves superior performance to baselines across the series of common sense reasoning and MMLU tasks.

Table 6: Comparison of perplexity on Wikitext, accuracy on eight 0-shot common sense reasoning tasks including ARC-challenge, ARC-easy, BoolQ, HellaSwag, Openbook QA, PIQA, SIQA, and WinoGrande, and 0-shot massive multitask language understanding tasks across four subjects: STEM, Humanities, Social Sciences, and MMLU-other, for the Llama 2, Llama 3 and Llama 3.2 families when quantized to W/A/KV = 4/4/4 bits. Results of all techniques were obtained using their official codebase. *: Mixed precision with $1/8$ channels in 8-bit for W/A/KV = 4.5-bit. All techniques except RTN use GPTQ Frantar et al. (2022). (↓): lower is better, (↑): higher is better.

Llama 2 family																	
Model	Method	Perplexity	0-shot common sense reasoning tasks									0-shot MMLU tasks				
		Wiki (↓)	ARC-c (↑)	ARC-e (↑)	BoolQ (↑)	HellaS (↑)	OBQA (↑)	PIQA (↑)	SIQA (↑)	WinoG (↑)	Avg. (↑)	humanities (↑)	Other (↑)	SocialS (↑)	STEM (↑)	Avg. (↑)
Llama-2-7b-hf	16-bit	5.5	46.3	74.6	77.8	75.9	44.2	79.2	46.1	69.1	64.1	38.9	45.9	46.0	33.4	41.1
	RTN	1766.2	26.3	27.8	54.8	29.4	25.8	51.0	35.0	48.7	37.4	24.5	24.7	22.9	22.2	23.6
	GPTQ	9600.0	24.8	31.4	55.4	30.6	25.6	55.8	34.2	53.3	38.9	24.7	24.5	22.7	23.2	23.8
	SmoothQuant+	15.4	29.3	47.1	56.8	48.6	31.8	65.5	37.2	52.4	46.1	25.0	24.5	24.1	23.4	24.2
	QUIK*	7.5	39.8	63.7	68.9	68.3	37.8	72.9	42.1	62.4	57.0	26.9	29.6	28.8	25.8	27.8
	QuaRot	6.1	41.5	71.4	73.2	73.2	40.6	76.9	43.6	65.6	60.7	31.2	35.1	34.6	28.2	32.3
	SpinQuant	6.0	43.6	71.3	73.8	73.2	40.4	76.0	**44.1**	65.4	61.0	33.9	38.5	37.5	29.5	34.8
	ResQ*	**5.8**	**44.0**	**72.6**	**75.3**	**74.0**	**41.0**	**77.9**	43.9	**66.9**	**62.0**	**35.9**	**40.9**	**42.2**	**32.2**	**37.7**
Llama-2-13b-hf	16-bit	4.9	49.1	77.4	80.5	79.4	45.2	80.7	47.2	72.1	66.5	47.9	59.3	61.0	42.4	52.7
	RTN	3543.9	22.8	29.8	40.2	26.6	27.8	51.4	35.6	50.6	33.5	23.7	25.0	23.1	22.6	23.6
	GPTQ	3120.0	23.6	31.1	38.7	27.2	26.8	53.6	35.8	49.8	33.8	25.0	25.4	23.7	25.1	24.8
	SmoothQuant+	11.2	34.5	55.6	62.9	62.5	32.4	70.1	38.7	55.6	51.0	25.7	26.1	27.3	27.3	26.6
	QUIK*	6.8	43.7	68.0	71.3	73.3	40.0	75.7	45.1	64.6	60.2	34.7	40.6	39.8	31.8	36.7
	QuaRot	5.4	46.9	74.9	76.6	75.8	42.6	79.1	45.5	69.0	63.8	43.8	53.6	54.0	39.4	47.7
	SpinQuant	5.2	49.0	**76.3**	78.2	77.1	42.8	**79.3**	46.3	69.5	64.8	43.5	53.1	55.4	39.1	47.8
	ResQ*	**5.1**	**49.1**	76.1	**79.7**	**77.9**	**43.6**	79.1	**46.6**	**69.9**	**65.2**	**45.3**	**56.0**	**58.0**	**41.0**	**50.1**

Llama 3 family																	
Model	Method	Perplexity	0-shot common sense reasoning tasks									0-shot MMLU tasks				
		Wiki (↓)	ARC-c (↑)	ARC-e (↑)	BoolQ (↑)	HellaS (↑)	OBQA (↑)	PIQA (↑)	SIQA (↑)	WinoG (↑)	Avg. (↑)	humanities (↑)	Other (↑)	SocialS (↑)	STEM (↑)	Avg. (↑)
Meta-Llama-3-8B	16-bit	6.1	53.2	77.1	81.1	79.2	44.8	80.9	47.0	73.4	67.1	55.0	70.6	73.2	53.7	63.1
	RTN	218.9	25.3	34.9	44.2	38.3	27.8	56.5	36.8	50.8	39.3	24.7	25.1	23.3	21.4	23.6
	GPTQ	166.3	24.7	37.7	44.3	36.8	27.0	57.6	36.4	53.8	39.8	24.7	23.9	22.8	21.8	23.3
	SmoothQuant+	78.2	27.5	42.0	50.7	44.9	28.8	59.0	35.9	50.9	42.5	25.4	25.5	24.5	23.4	24.7
	QUIK*	14.2	33.6	56.4	60.5	61.5	33.2	68.7	39.9	59.0	51.6	30.0	34.0	34.8	32.1	32.7
	QuaRot	7.8	45.1	70.4	73.8	74.7	42.6	76.6	45.1	68.5	62.1	47.8	59.1	61.4	44.3	53.2
	SpinQuant	7.4	48.0	**75.4**	**75.8**	75.4	**43.8**	77.5	45.0	69.2	63.8	49.8	63.3	65.0	46.8	56.2
	ResQ*	**7.1**	**49.2**	75.0	72.5	**76.5**	43.0	**78.3**	**45.8**	**71.0**	**63.9**	**50.6**	**64.4**	**65.8**	**48.1**	**57.2**
Meta-Llama-3-70B	16-bit	2.9	64.2	85.9	85.3	84.9	48.6	84.4	50.8	80.6	73.1	67.6	81.5	86.8	68.4	76.1
	RTN	432.7	32.6	50.3	54.2	41.3	31.6	64.8	35.9	53.2	45.5	24.5	23.8	22.3	22.1	23.2
	GPTQ	11655.0	25.9	26.0	37.9	26.2	28.6	50.4	34.3	49.9	34.9	27.1	24.3	24.0	26.5	25.5
	SmoothQuant+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	QUIK*	8.0	44.5	68.9	60.7	75.0	36.4	76.1	43.2	60.4	58.2	46.6	56.4	58.0	43.6	51.1
	QuaRot	5.7	53.7	74.5	81.6	81.1	**46.6**	81.0	46.8	75.2	67.6	55.7	72.5	75.8	57.3	65.3
	SpinQuant	6.2	52.0	77.3	81.7	75.6	43.8	78.8	43.4	72.8	65.7	50.7	67.0	68.1	51.9	59.4
	ResQ*	**4.1**	**61.4**	**84.3**	**83.9**	**83.5**	46.0	**83.1**	**48.6**	**78.3**	**71.1**	**64.9**	**79.9**	**84.9**	**66.1**	**74.0**

Llama 3.2 family																	
Model	Method	Perplexity	0-shot common sense reasoning tasks									0-shot MMLU tasks				
		Wiki (↓)	ARC-c (↑)	ARC-e (↑)	BoolQ (↑)	HellaS (↑)	OBQA (↑)	PIQA (↑)	SIQA (↑)	WinoG (↑)	Avg. (↑)	humanities (↑)	Other (↑)	SocialS (↑)	STEM (↑)	Avg. (↑)
Llama-3.2-1B	16-bit	9.8	36.5	60.6	63.4	63.6	37.4	74.5	42.8	60.1	54.9	34.8	41.1	39.9	32.0	36.9
	RTN	329.1	22.4	29.9	53.4	31.4	29.4	54.8	34.9	48.5	38.1	24.8	25.2	22.4	22.7	23.8
	GPTQ	108.9	24.7	32.7	52.3	30.7	23.6	54.3	34.4	51.1	38.0	24.7	25.1	25.5	24.5	24.9
	SmoothQuant+	228.9	23.3	30.1	52.9	31.3	26.6	54.2	34.5	51.2	38.0	23.9	24.1	25.0	23.5	24.1
	QUIK*	21.8	27.4	46.0	55.0	46.0	26.4	62.4	38.6	52.6	44.3	25.6	25.6	24.6	24.5	25.1
	QuaRot	14.3	30.0	51.4	59.1	54.0	**34.2**	66.7	39.6	57.1	49.0	25.4	26.9	25.4	24.4	25.5
	SpinQuant	13.6	32.3	51.8	**59.3**	55.4	30.4	67.7	38.6	54.7	48.8	25.4	27.6	24.2	25.3	25.6
	ResQ*	**12.4**	**34.0**	**54.2**	57.0	**57.3**	31.2	**69.4**	**41.0**	**56.8**	**50.1**	**28.3**	**30.5**	**31.3**	**27.6**	**29.4**
Llama-3.2-3B	16-bit	7.8	46.2	71.7	73.1	73.7	43.4	77.4	47.2	69.1	62.7	48.9	62.9	62.3	45.2	54.8
	RTN	268.8	23.5	35.4	46.2	35.6	28.2	56.3	33.6	50.6	38.7	25.1	25.6	27.0	24.9	25.7
	GPTQ	178.3	27.0	27.0	48.8	44.4	27.8	59.1	37.1	51.5	40.3	24.9	24.5	25.7	24.0	24.8
	SmoothQuant+	96.1	25.3	33.1	47.8	37.7	25.2	56.2	35.8	50.9	39.0	25.4	26.6	26.4	25.3	25.9
	QUIK*	15.8	32.9	50.1	52.6	59.1	33.2	68.7	40.3	53.0	48.8	29.0	33.2	31.9	30.3	31.1
	QuaRot	10.1	38.6	59.0	65.9	66.5	35.8	74.4	43.1	**65.2**	56.1	38.5	47.3	46.7	35.3	42.0
	SpinQuant	9.2	38.9	64.8	68.0	69.1	**39.4**	74.9	45.1	62.9	57.9	37.0	49.4	50.5	39.9	44.2
	ResQ*	**8.8**	**43.1**	**65.6**	**68.8**	**70.5**	38.4	**75.1**	**45.6**	64.8	**59.0**	**44.7**	**57.0**	**56.5**	**41.0**	**49.8**

Table 7: Comparison of perplexity score on Wikitext, accuracy on eight 0-shot common sense reasoning tasks including ARC-challenge, ARC-easy, BoolQ, HellaSwag, Openbook QA, PIQA, SIQA, and WinoGrande, and 0-shot massive multitask language understanding tasks across four subjects: STEM, Humanities, Social Sciences, and MMLU-other, for the Qwen2.5 family when quantized to W/A/KV = 4/4/4 bits. Results of all techniques were obtained using their official codebase. *: Mixed precision with $1/8$ channels in 8-bit for W/A/KV = 4.5-bit. All techniques except RTN use GPTQ Frantar et al. (2022). (↓): lower is better, (↑): higher is better.

Qwen2.5 family																	
Model	Method	Perplexity	0-shot common sense reasoning tasks									0-shot MMLU tasks				
		Wiki (↓)	ARC-c (↑)	ARC-e (↑)	BoolQ (↑)	HellaS (↑)	OBQA (↑)	PIQA (↑)	SIQA (↑)	WinoG (↑)	Avg. (↑)	humanities (↑)	Other (↑)	SocialS (↑)	STEM (↑)	Avg. (↑)
Qwen2.5-0.5B	16-bit	13.1	31.9	58.4	62.1	52.1	35.0	69.7	44.3	57.1	51.3	42.2	53.2	55.5	41.5	48.1
	RTN	23204.3	26.2	27.0	39.3	26.0	24.0	50.7	34.5	51.5	34.9	24.8	24.0	22.8	24.3	23.9
	GPTQ	16302.3	23.7	26.9	39.0	26.5	26.4	50.2	33.4	49.6	34.5	24.1	24.8	23.5	23.0	23.9
	SmoothQuant+	10053.9	25.9	26.3	39.9	27.2	25.4	47.1	35.9	49.6	34.7	24.5	24.7	21.5	22.1	23.2
	QUIK*	38.6	24.5	38.6	48.0	36.9	28.4	58.1	**36.4**	**51.9**	40.4	**26.3**	25.9	23.6	24.2	25.0
	QuaRot	219.9	25.4	36.6	45.0	28.9	**28.6**	54.1	32.9	51.7	37.9	24.4	24.0	23.0	23.5	23.7
	ResQ*	**29.6**	**27.1**	**44.2**	**53.2**	**38.8**	28.0	**61.9**	34.4	51.3	**42.4**	26.1	**27.5**	**25.3**	**26.0**	**26.2**
Qwen2.5-1.5B	16-bit	9.3	45.1	72.1	72.9	67.7	40.2	76.3	48.8	63.7	60.8	53.5	65.5	70.6	52.8	60.6
	RTN	14518.9	23.1	27.2	43.9	26.8	25.6	51.3	33.4	52.5	35.5	23.8	24.5	23.8	22.7	23.7
	GPTQ	25769.7	23.9	26.9	43.9	26.1	27.6	49.7	32.1	51.5	35.2	24.6	24.7	23.7	23.8	24.2
	SmoothQuant+	31655.9	25.0	26.2	39.9	26.0	26.0	50.8	32.1	49.0	34.4	25.5	24.4	22.7	22.4	23.8
	QUIK*	6613.5	21.8	31.9	40.9	27.9	27.4	52.8	35.2	48.6	35.8	24.6	24.0	21.9	21.7	23.1
	QuaRot	6599.9	23.6	37.3	46.2	28.6	27.0	56.3	35.2	52.4	38.3	24.5	24.3	23.0	22.4	23.5
	ResQ*	**12.5**	**38.7**	**64.1**	**65.7**	**61.4**	**37.8**	**71.6**	**42.7**	**60.1**	**55.3**	**43.2**	**54.4**	**54.9**	**41.5**	**48.5**
Qwen2.5-3B	16-bit	8.0	47.4	73.0	77.5	73.6	42.0	78.7	49.9	68.4	63.8	56.6	71.0	76.3	60.6	66.1
	RTN	39033.0	25.6	25.8	41.7	26.3	27.4	49.5	33.1	51.4	35.1	24.5	24.4	22.8	21.9	23.4
	GPTQ	9977.8	26.0	26.7	41.5	26.7	28.2	51.5	31.9	48.3	35.1	24.3	23.8	22.8	21.8	23.2
	SmoothQuant+	73306.7	25.4	24.5	41.0	26.4	29.8	48.4	32.4	50.4	34.8	25.6	24.7	23.1	22.4	23.9
	QUIK*	15.5	36.1	55.4	61.4	57.2	36.2	67.1	40.8	55.3	51.2	36.4	42.8	42.4	36.1	39.4
	QuaRot	68.8	32.4	53.1	51.6	49.2	33.4	66.7	39.3	56.4	47.7	28.1	32.0	28.9	26.6	28.9
	ResQ*	**9.0**	**45.3**	**70.5**	**72.7**	**70.2**	**42.4**	**76.8**	**46.7**	**64.4**	**61.1**	**53.1**	**66.5**	**70.5**	**54.8**	**61.2**
Qwen2.5-7B	16-bit	6.8	51.2	77.6	84.7	78.9	47.2	80.0	54.8	73.2	68.4	62.6	76.7	82.6	70.1	73.0
	RTN	24382.1	24.5	26.3	37.8	26.0	29.0	51.0	34.1	50.1	34.9	24.9	24.3	23.4	24.9	24.4
	GPTQ	13593.7	25.2	25.6	37.8	26.3	28.2	52.4	34.4	48.9	34.8	24.4	24.3	22.8	22.6	23.5
	SmoothQuant+	19088.7	26.3	25.2	39.8	26.4	27.6	52.7	33.5	52.0	35.4	25.1	25.4	22.6	24.1	24.3
	QUIK*	260.3	29.5	42.4	51.7	36.3	28.2	59.6	34.5	49.6	41.5	24.3	26.9	23.1	23.8	24.6
	QuaRot	4035.9	25.9	41.0	39.1	29.1	27.6	57.9	35.7	50.6	38.4	24.8	24.4	24.4	22.7	24.1
	ResQ*	**8.2**	**49.0**	**74.7**	**81.4**	**75.7**	**45.0**	**78.9**	**49.4**	**68.2**	**65.3**	**57.8**	**74.4**	**79.3**	**64.5**	**69.0**
Qwen2.5-14B	16-bit	5.3	58.8	79.4	85.4	82.9	45.4	81.9	55.3	75.8	70.6	69.9	81.9	86.2	76.5	78.6
	RTN	2715	21.6	32.7	51.5	29.6	25.8	52.6	33.2	51.7	37.3	25.3	23.2	26.0	25.3	24.9
	GPTQ	5100.3	23.8	29.1	47.7	30.1	27.6	51.3	34.6	51.2	36.9	25.1	24.7	25.1	24.3	24.8
	SmoothQuant+	1375.7	27.0	26.3	38.0	26.8	29.2	51.6	32.4	49.3	35.1	25.9	24.5	22.2	22.2	23.7
	QUIK*	10.5	45.0	67.1	64.7	68.9	37.6	74.8	43.9	59.3	57.6	48.9	61.1	64.7	51.5	56.6
	QuaRot	6.8	54.8	79.6	79.9	78.7	44.0	79.5	49.9	**70.7**	67.1	60.9	75.1	80.2	67.3	70.9
	ResQ*	**6.2**	**57.6**	**82.1**	**84.9**	**81.1**	**44.8**	**80.5**	**51.7**	70.6	**69.2**	**65.2**	**78.4**	**83.4**	**71.5**	**74.6**
Qwen2.5-32B	16-bit	5.0	55.7	78.0	87.4	84.1	44.4	82.3	56.4	75.2	70.4	73.1	83.6	89.6	81.2	81.9
	RTN	1847.4	24.3	35.3	51.4	31.9	27.0	52.8	34.1	51.4	38.5	24.5	25.1	25.3	24.3	24.8
	GPTQ	3891.1	25.4	35.4	48.5	31.8	27.0	53.8	35.8	50.5	38.5	25.9	24.8	23.6	24.0	24.6
	SmoothQuant+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	QUIK*	9.6	41.0	64.6	74.9	72.0	39.6	75.8	44.5	60.2	59.1	54.7	66.8	71.3	58.8	62.9
	QuaRot	6.1	54.5	76.1	85.1	81.5	44.2	80.1	51.3	70.4	67.9	68.5	80.0	86.0	76.0	77.6
	ResQ*	**5.6**	**55.1**	**78.4**	**86.0**	**82.5**	**45.4**	**81.1**	**53.9**	**74.0**	**69.5**	**70.3**	**82.3**	**87.9**	**78.9**	**79.8**
Qwen2.5-72B	16-bit	3.9	62.6	83.2	89.2	86.0	46.6	83.6	58.4	77.7	73.4	77.2	86.9	90.6	82.4	84.3
	RTN	45412.7	25.9	26.3	38.0	25.9	25.2	50.0	34.2	48.7	34.3	25.5	24.2	23.0	23.2	24.0
	GPTQ	37967.2	25.4	25.8	38.1	25.6	26.6	51.2	34.2	49.4	34.5	25.1	24.0	21.9	22.2	23.3
	SmoothQuant+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	QUIK*	8.3	45.1	68.1	77.2	77.2	39.0	77.4	45.6	65.6	61.9	60.2	74.3	77.5	65.3	69.3
	QuaRot	4.9	55.8	**81.1**	87.5	84.0	45.2	81.7	52.5	74.5	70.3	71.4	84.2	87.7	77.1	80.1
	ResQ*	**4.6**	**58.4**	80.9	**88.4**	**84.9**	**48.2**	**82.6**	**55.5**	**77.0**	**72.0**	**72.8**	**84.6**	**89.0**	**79.5**	**81.5**

Table 8: Accuracy (higher is better) on 0-shot massive multi-discipline multimodal understanding and reasoning tasks across six subjects: Art & Design, Business, Science, Health & Medicine, Humanities & Social Science, and Tech & Engineering for the Qwen2 VL Instruct family. Results of all techniques were obtained using their official codebase. Our work ResQ and QUIK Ashkboos et al. (2024b) keep $1/8$ of channels in 8-bit for average of 4.5-bit. All techniques except RTN use GPTQ Frantar et al. (2022).

Qwen2-VL-2B-Instruct								
W/A/KV (bit)	Method	0-shot MMMU tasks						
		Art-Design	Business	Science	Health	Humanities	Tech	Avg.
16/16/16	Baseline	56.7	36.0	37.3	50.8	26.0	31.0	39.6
4/4/4	RTN	28.3	18.7	26.0	26.7	21.3	**29.1**	25.0
	GPTQ	28.3	**27.3**	27.0	29.0	**26.7**	27.6	27.7
	QuaRot	24.2	23.3	20.7	26.7	26.0	22.9	24.0
4.5/4.5/4.5	QUIK	25.8	26.0	26.7	29.2	26.0	24.3	26.3
	ResQ	**38.3**	21.3	**28.7**	**45.0**	21.3	23.3	**29.7**
4/8/4	RTN	27.5	21.3	27.3	24.2	21.3	27.6	24.9
	GPTQ	24.2	23.3	24.0	18.3	21.3	29.5	23.4
	QuaRot	20.0	24.7	30.0	26.7	26.0	**31.4**	26.5
4.5/8/4.5	QUIK	33.3	28.7	32.0	32.5	26.0	18.1	28.4
	ResQ	**37.5**	**32.0**	**32.7**	**47.5**	**26.7**	27.6	**34.0**
Qwen2-VL-7B-Instruct								
W/A/KV (bit)	Method	0-shot MMMU tasks						
		Art-Design	Business	Science	Health	Humanities	Tech	Avg.
16/16/16	Baseline	68.3	41.3	54.7	68.3	38.7	38.1	51.6
4/4/4	RTN	24.2	28.0	29.3	22.5	29.3	27.1	26.7
	GPTQ	21.7	26.0	25.3	28.3	24.7	23.3	24.9
	QuaRot	21.7	21.3	28.7	25.0	20.7	29.5	24.5
4.5/4.5/4.5	QUIK	30.8	30.0	32.0	26.7	28.0	26.2	28.9
	ResQ	**65.0**	**39.3**	**45.3**	**61.7**	**34.0**	**36.7**	**47.0**
4/8/4	RTN	23.3	28.7	27.3	25.0	22.7	24.3	25.2
	GPTQ	20.8	23.3	30.0	19.2	24.0	28.6	24.3
	QuaRot	20.8	26.0	30.0	19.2	24.7	26.2	24.5
4.5/8/4.5	QUIK	25.0	23.3	31.3	26.7	25.3	26.7	26.4
	ResQ	**67.5**	**39.3**	**51.3**	**64.2**	**36.7**	**33.8**	**48.8**

Table 9: Comparison of perplexity score on Wikitext, accuracy on eight 0-shot common sense reasoning tasks including ARC-challenge, ARC-easy, BoolQ, HellaSwag, Openbook QA, PIQA, SIQA, and WinoGrande, and 0-shot massive multitask language understanding tasks across four subjects: STEM, Humanities, Social Sciences, and MMLU-other, for the Qwen2.5 family when quantized to W/A/KV = 4/4/4 bits. (↓): lower is better, (↑): higher is better.

Qwen2.5 family																
Model	Method	Perplexity	0-shot common sense reasoning tasks									0-shot MMLU tasks				
		Wiki	ARC-c	ARC-e	BoolQ	HellaS	OBQA	PIQA	SIQA	WinoG	Avg.	humanities	Other	SocialS	STEM	Avg.
		(↓)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)	(↑)
Qwen2.5-1.5B	QuaRot	6599.9	23.6	37.3	46.2	28.6	27.0	56.3	35.2	52.4	38.3	24.5	24.3	23.0	22.4	23.5
	SpinQuant	5938.3	22.8	36.4	47.3	29.0	26.2	56.4	35.4	51.3	38.1	25.0	24.1	23.3	22.5	23.7
	ResQ	**187.3**	**31.1**	**54.0**	**54.7**	**43.6**	**33.2**	**64.8**	**39.1**	**52.4**	**46.6**	**26.3**	**30.3**	**27.8**	**24.5**	**27.2**
Qwen2.5-3B	QuaRot	68.8	32.4	53.1	51.6	49.2	33.4	66.7	39.3	56.4	47.7	28.1	32.0	28.9	26.6	28.9
	SpinQuant	70.6	33.3	56.4	51.8	49.5	33.0	66.7	41.0	57.1	48.6	30.1	37.2	33.9	30.1	32.8
	ResQ	**9.8**	**44.1**	**70.2**	**70.2**	**67.3**	**41.0**	**73.7**	**45.4**	**61.1**	**59.1**	**45.8**	**58.4**	**58.9**	**45.9**	**52.2**
Qwen2.5-7B	QuaRot	4035.9	25.9	41.0	39.1	29.1	27.6	57.9	35.7	50.6	38.4	24.8	24.4	24.4	22.7	24.1
	SpinQuant	3395.4	25.5	44.2	39.5	29.3	27.0	58.8	35.3	49.4	38.6	25.0	23.8	24.2	24.3	24.3
	ResQ	**34.2**	**42.6**	**60.2**	**75.4**	**59.4**	**41.6**	**63.1**	**44.6**	**62.8**	**56.2**	**49.3**	**63.6**	**68.6**	**50.4**	**58.0**
Qwen2.5-14B	QuaRot	6.8	54.8	79.6	79.9	78.7	44.0	79.5	49.9	**70.7**	67.1	60.9	75.1	**80.2**	67.3	70.9
	SpinQuant	6.6	54.2	81.2	82.1	77.9	**44.6**	78.4	50.8	70.0	67.4	59.6	**75.3**	78.9	66.7	70.1
	ResQ	**6.5**	**55.6**	**81.3**	**89.2**	**79.2**	43.6	**79.9**	**51.4**	70.2	**67.5**	**61.6**	**75.3**	80.1	**68.2**	**71.3**
Qwen2.5-32B	QuaRot	6.1	54.3	78.6	83.0	81.0	43.4	79.8	50.6	71.7	67.8	67.4	80.5	85.2	75.1	77.0
	SpinQuant	6.0	54.5	76.1	85.1	81.5	44.2	80.1	51.3	70.4	67.9	**68.5**	80.0	86.0	76.0	77.6
	ResQ	**5.9**	**55.9**	**79.7**	**85.8**	**81.6**	**44.8**	**80.3**	**51.6**	**73.4**	**69.1**	68.4	**80.6**	**86.4**	**76.2**	**77.9**
Qwen2.5-72B	QuaRot	**4.9**	55.8	81.1	**87.5**	84.0	45.2	81.7	52.5	74.5	70.3	**71.4**	**84.2**	**87.7**	77.1	**80.1**
	SpinQuant	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	ResQ	**4.9**	**58.8**	**81.9**	87.2	**84.1**	**46.0**	**82.2**	**53.2**	**75.5**	**71.1**	71.3	**84.2**	87.4	**77.4**	**80.1**