Memory Efficient Optimizers with 4-bit States

Bingrui Li, Jianfei Chen, Jun Zhu

Tsinghua University

Background

Data memory

- input data and activation in each layer

Model memory

- model parameters, optimizer states (and gradients)

Other (temporary) memory

- GPU kernel, cache, etc.

Background

$$\mathbf{Adam}(\mathbf{w}_{t-1}, \mathbf{m}_{t-1}, \mathbf{v}_{t-1}, \mathbf{g}_t) = \begin{cases} \mathbf{m}_t & = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\ \mathbf{v}_t & = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \\ \hat{\mathbf{m}}_t & = \mathbf{m}_t/(1-\beta_1^t) \\ \hat{\mathbf{v}}_t & = \mathbf{v}_t/(1-\beta_2^t) \\ \mathbf{w}_t & = \mathbf{w}_{t-1} - \eta \cdot \hat{\mathbf{m}}_t / \left(\sqrt{\hat{\mathbf{v}}_t} + \epsilon\right) \end{cases}$$

Data memory

- input data and activation in each layer

Model memory

- model parameters, optimizer states (and gradients)

Other (temporary) memory

- GPU kernel, cache, etc.

For LLaMA-7B:

- number of parameters: 7B
- number of optimizer states: 14B when fine-tuned with AdamW (32-bit),
- memory of optimizer states: about 52.2GB.

Goal: Reduce the memory consumption of optimizer states (in stateful optimizers), especially AdamW

Memory Efficient Methods

On optimizer states:

- Quantization-based: 8-bit Adam (Dettmers et al. ICLR 2022)
- Factorization-based: Adafactor (Shazeer et al. ICML 2018), SM3 (Anil et al. NeurIPS 2019), Extreme Tensoring (Chen et al. ICLR 2020)
- By tuning fewer parameters: LoRA (Hu et al. ICLR 2022), prefix tuning (Li et al. 2021) etc.

Factorization-based Method: Adafactor

Shazeer et al. ICML 2018

$$\underset{R\in\mathbb{R}^{n\times k}, S\in\mathbb{R}^{k\times m}}{\text{minimize}} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} d(V_{ij}, [RS]_{ij})$$

$$\text{subject to} \quad R_{ij} \geq 0, S_{ij} \geq 0.$$

$$R = V1_m, \quad S = \frac{1_n^\top V}{1_n^\top V 1_m}.$$

Algorithm 2 Adam for a matrix parameter X with factored second moments and first moment decay parameter $\beta_1 = 0$.

1: **Inputs:** initial point $X_0 \in \mathbb{R}^{n\times m}$, step sizes $\{\alpha_t\}_{t=1}^T$, second moment decay β_2, regularization constant ϵ

2: Initialize $R_0 = 0$ and $C_0 = 0$

3: **for** $t = 1$ **to** T **do**

4: $\quad G_t = \nabla f_t(X_{t-1})$

5: $\quad R_t = \beta_2 R_{t-1} + (1-\beta_2)(G_t^2)1_m$

6: $\quad C_t = \beta_2 C_{t-1} + (1-\beta_2)1_n^\top(G_t^2)$

7: $\quad \hat{V}_t = (R_t C_t / 1_n^\top R_t)/(1-\beta_2^t)$

8: $\quad X_t = X_{t-1} - \alpha_t G_t / (\sqrt{\hat{V}_t} + \epsilon)$

9: **end for**

Factorization-based Method: Adafactor

Shazeer et al. ICML 2018

$$\underset{R\in\mathbb{R}^{n\times k}, S\in\mathbb{R}^{k\times m}}{\text{minimize}} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} d(V_{ij}, [RS]_{ij})$$

$$\text{subject to} \quad R_{ij}\geq 0, S_{ij}\geq 0.$$

$$R = V1_m, \quad S = \frac{1_n^\top V}{1_n^\top V 1_m}.$$

Algorithm 2 Adam for a matrix parameter X with factored second moments and first moment decay parameter $\beta_1 = 0$.

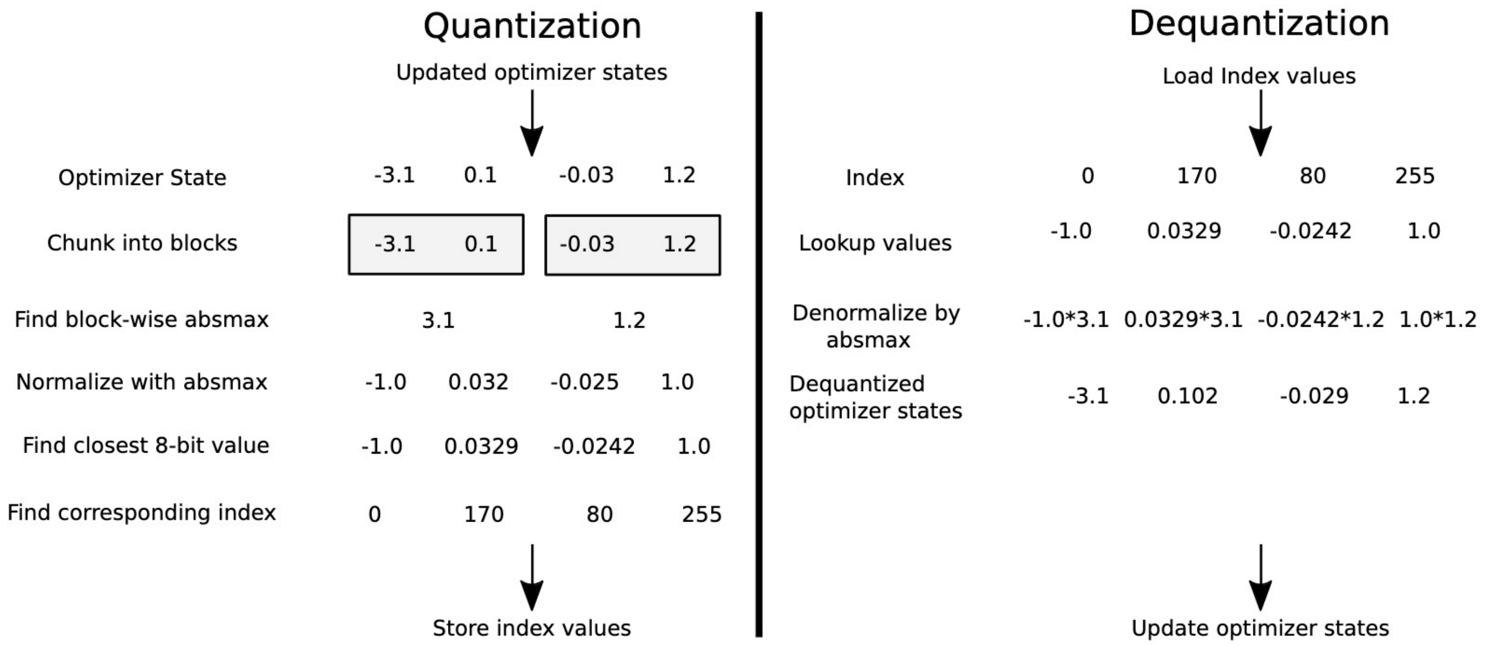
Inputs: initial point $X_0 \in \mathbb{R}^{n\times m}$, step sizes $\{\alpha_t\}_{t=1}^{T}$, second moment decay β_2, regularization constant ϵ
Initialize $R_0 = 0$ and $C_0 = 0$
for $t = 1$ **to** T **do**
    $G_t = \nabla f_t(X_{t-1})$
    $R_t = \beta_2 R_{t-1} + (1-\beta_2)(G_t^2)1_m$
    $C_t = \beta_2 C_{t-1} + (1-\beta_2)1_n^\top(G_t^2)$
    $\hat{V}_t = (R_t C_t / 1_n^\top R_t)/(1-\beta_2^t)$
    $X_t = X_{t-1} - \alpha_t G_t/(\sqrt{\hat{V}_t} + \epsilon)$
end for

However, this can only apply to second moments

Quantzation-based Method

Dettmers et al. ICLR 2022

$$\mathbf{Adam}(\mathbf{w}_{t-1}, \mathbf{m}_{t-1}, \mathbf{v}_{t-1}, \mathbf{g}_t) = \begin{cases} \mathbf{m}_t & = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\ \mathbf{v}_t & = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \\ \hat{\mathbf{m}}_t & = \mathbf{m}_t/(1-\beta_1^t) \\ \hat{\mathbf{v}}_t & = \mathbf{v}_t/(1-\beta_2^t) \\ \mathbf{w}_t & = \mathbf{w}_{t-1} - \eta \cdot \hat{\mathbf{m}}_t / \left(\sqrt{\hat{\mathbf{v}}_t} + \epsilon\right) \end{cases}$$



Memory Efficient Methods

Our 4-bit AdamW

On optimizer states:

- Quantization-based: 8-bit Adam (Dettmers et al. ICLR 2022)
- Factorization-based: Adafactor (Shazeer et al. ICML 2018), SM3 (Anil et al. NeurIPS 2019), Extreme Tensoring (Chen et al. ICLR 2020)
- By tuning fewer parameters: LoRA (Hu et al. ICLR 2022), prefix tuning (Li et al. 2021) etc.

Preliminaries: Quantization

Disentangle quantizer Q(·): normalization N(·) and mapping M(·)

$$q_j := \mathbf{Q}(x_j) = \mathbf{M} \circ \mathbf{N}(x_j).$$

Preliminaries: Quantization

Disentangle quantizer Q(·): normalization N(·) and mapping M(·)

$$q_j := \mathbf{Q}(x_j) = \mathbf{M} \circ \mathbf{N}(x_j).$$

Normalization: scale each elements of original tensor into the unit interval

$$n_j := \mathbf{N}_{\text{per-tensor}}(x_j) = x_j / \max_{1 \le i \le p} |x_i| \,,$$
$$n_j := \mathbf{N}_{\text{block-wise}}(x_j) = x_j / \max \left\{ |x_i| : 1 + B \lfloor j/B \rfloor \le i \le B \left(\lfloor j/B \rfloor + 1 \right) \right\},$$

Different normalization methods give different quantization error and memory overhead.

Preliminaries: Quantization

Disentangle quantizer Q(·): normalization N(·) and mapping M(·)

$$q_j := \mathbf{Q}(x_j) = \mathbf{M} \circ \mathbf{N}(x_j).$$

Mapping: convert normalized tensors to low-bit integers.

Given a predefined map **T**: $[0, 2^b - 1] \cap Z \rightarrow [0, 1]$

$$q_j := \mathbf{M}(n_j) = \arg \min_{0 \leq i < 2^b} |n_j - \mathbf{T}(i)| \,.$$

Mapping gives nonlinearity to quantization. The design of **T** is crucial as it could effectively mitigate quantization error.

Quantization

Quantization Mapping: Linear and DE (dynamic exponent)

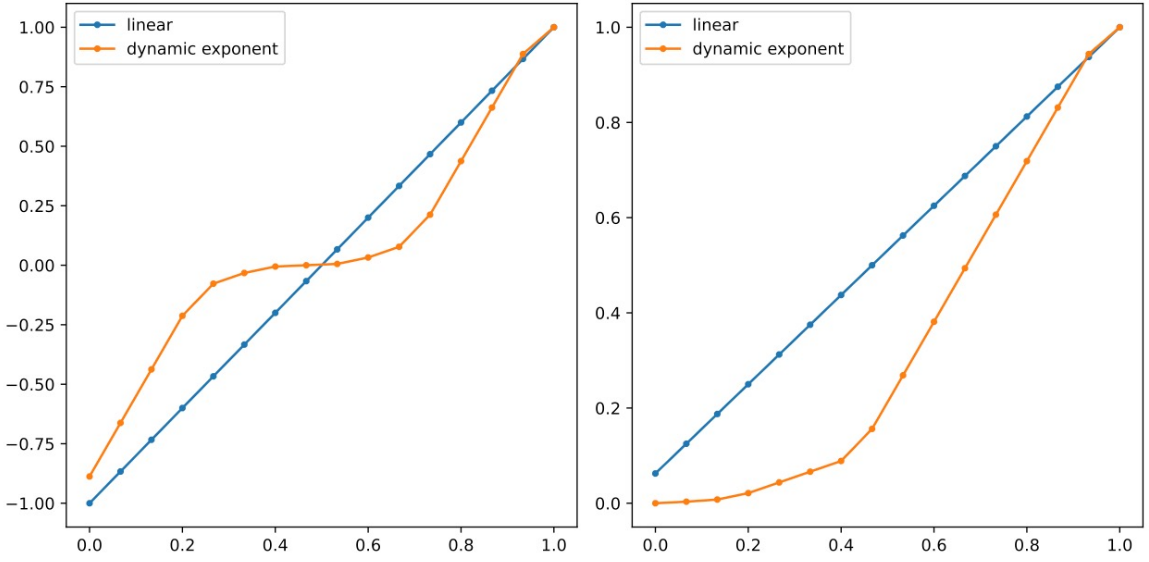


Figure 32: Visualization of the quantization mappings for the linear and dynamic exponent at 4-bit precision. Left: Signed case. Right: Unsigned case.

Notation: We use Norm./Map. to denote quantization methods, e.g., B2048/DE

Compressing First Moment

Observation: complicated outlier patterns

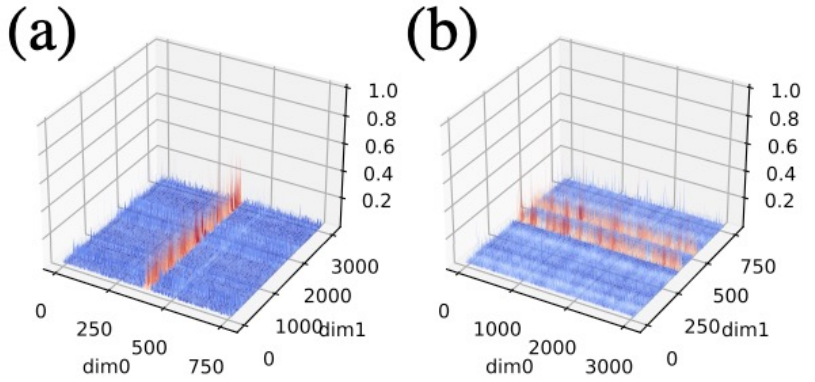


Figure 2: Outlier patterns vary across two first moment tensors. (a): outliers lie in fixed rows (dimension 0). (b): outliers lie in fixed columns (dimension 1).

Compressing First Moment

Smaller block size of 128 consistently (i.e., B128/DE)

enhance performance and **keep overhead under control**

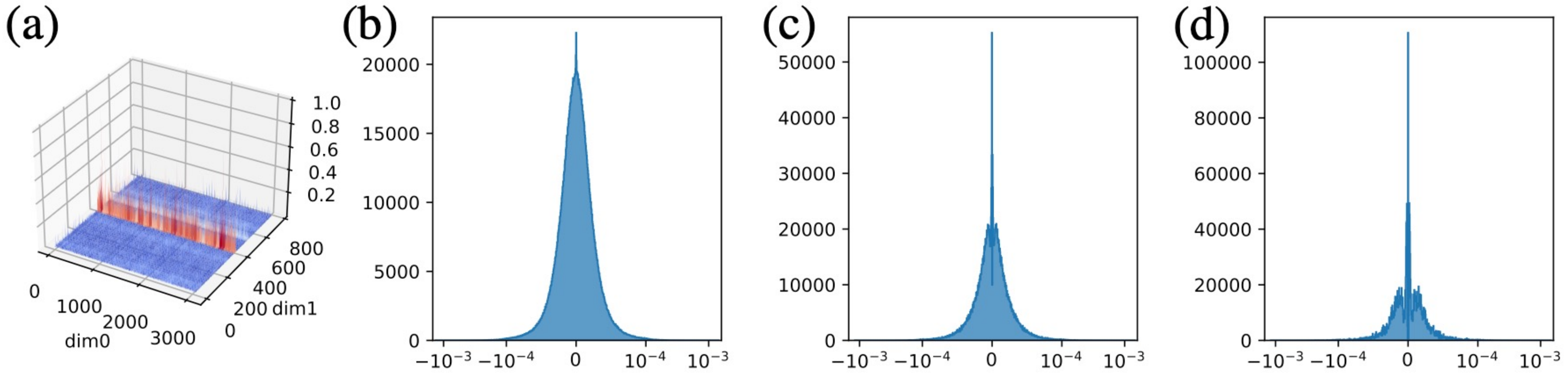


Figure 1: Visualization of the first moment in the `layers.3.blocks.1.mlp.fc1` layer in a Swin-T model. (a): Magnitude of the first moment. (b): Histogram of the first moment. (c): Moment approximated by `B128/DE`. (d): Moment approximated by `B2048/DE`.

Compressing Second Moment

Main bottleneck: **zero-point problem**

Empirically, zero is often the most frequent element in quantization.

But for Adam second moment, zero-point causes crash:

$$\mathbf{Adam}(\mathbf{w}_{t-1}, \mathbf{m}_{t-1}, \mathbf{v}_{t-1}, \mathbf{g}_t) = \begin{cases} \mathbf{m}_t & = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\ \mathbf{v}_t & = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \\ \hat{\mathbf{m}}_t & = \mathbf{m}_t / (1-\beta_1^t) \\ \hat{\mathbf{v}}_t & = \mathbf{v}_t / (1-\beta_2^t) \\ \mathbf{w}_t & = \mathbf{w}_{t-1} - \eta \cdot \hat{\mathbf{m}}_t / \left(\sqrt{\hat{\mathbf{v}}_t} + \epsilon\right) \end{cases}$$

w. Transform: $h(v) = 1 / (\sqrt{v} + 10^{-6})$

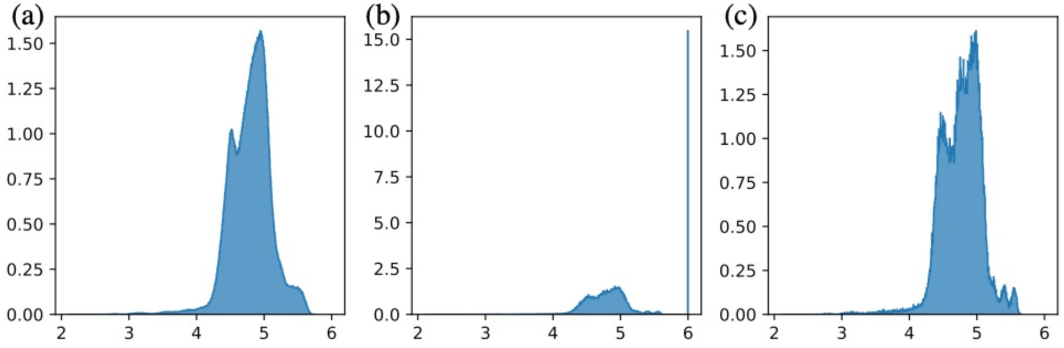


Figure 3: Histogram of the inverse square root of second moment. (a) full-precision; (b) quantized with B128/DE; (c) quantized with B128/DE-0. All figures are at log10 scale and y-axis represents density.

Compressing Second Moment

Main bottleneck: **zero-point problem**

Empirically, zero is often the most frequent element in quantization.

But for Adam second moment, zero-point causes crash:

$$
\mathbf{Adam}(\mathbf{w}_{t-1}, \mathbf{m}_{t-1}, \mathbf{v}_{t-1}, \mathbf{g}_t) = \begin{cases} \mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\ \mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2 \\ \hat{\mathbf{m}}_t &= \mathbf{m}_t / (1-\beta_1^t) \\ \hat{\mathbf{v}}_t &= \mathbf{v}_t / (1-\beta_2^t) \\ \mathbf{w}_t &= \mathbf{w}_{t-1} - \eta \cdot \hat{\mathbf{m}}_t / \left(\sqrt{\hat{\mathbf{v}}_t} + \epsilon\right) \end{cases}
$$

w. Transform: $h(v) = 1\,/(\sqrt{v} + 10^{-6})$

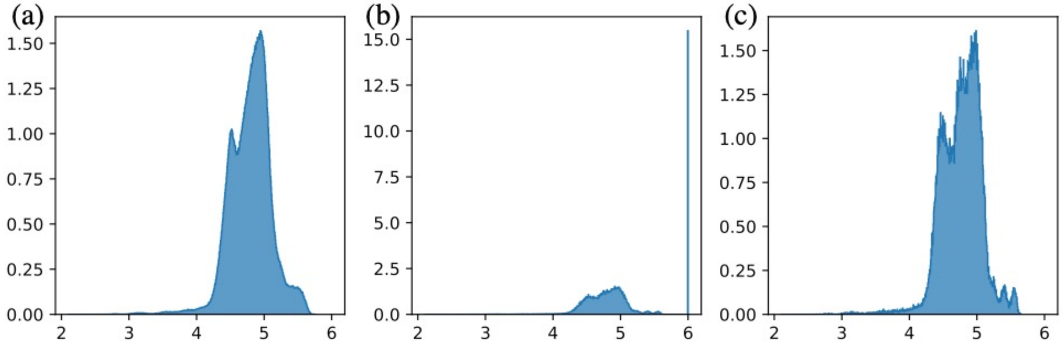


Figure 3: Histogram of the inverse square root of second moment. (a) full-precision; (b) quantized with B128/DE; (c) quantized with B128/DE-0. All figures are at log10 scale and y-axis represents density.

Approach:

1. remove zero in quantization map
2. factorization

Compressing Second Moment

New normalization method: **rank-1 normalization**

$$\mathbf{x} \in \mathbb{R}^{n \times m}$$

$$\mathbf{r} \in \mathbb{R}^n \text{ and } \mathbf{c} \in \mathbb{R}^m$$
$$r_i = \max_{1 \le j \le m} x_{i,j}$$
$$c_j = \max_{1 \le i \le n} x_{i,j}$$

$$\mathbf{N}_{\text{rank-1}}(x_{i,j}) = \frac{1}{\min\{r_i, c_j\}} x_{i,j}.$$

Pros

- deals with the outliers more smartly and effectively
- better memory efficiency and scalable to high dim tensors
- good performance

Cons

- cannot apply to 1-dim tensors and/or shape information is not available

Compressing Second Moment

Ablation Experiments

Table 1: Ablation analysis of 4-bit optimizers on the second moment on the GPT-2 Medium E2E-NLG finetuning task. The first line barely turns 8-bit Adam [15] into 4-bit, i.e. B2048/DE for both first and second moments. We only vary the quantization scheme for second moment. SR=stochastic rounding (see App. E.3 for details). Stable Embedding layers are not quantized. 32-bit AdamW achieves a BLEU of 67.7.

Normalization	Mapping	Stable Embed.*	Factorized	Unstable(%)	BLEU
B2048	DE	✗	✗	33	66.6 ± 0.61
B2048	DE	✓	✗	0	66.9 ± 0.52
B128	DE	✗	✗	66	$65.7 \pm$ N/A
B128	DE+SR*	✗	✗	33	65.4 ± 0.02
B128	DE	✓	✗	0	67.2 ± 1.13
B2048	DE-0	✗	✗	0	67.5 ± 0.97
B2048	DE-0	✓	✗	0	67.1 ± 1.02
B128	DE-0	✗	✗	0	67.4 ± 0.59
Rank-1	DE-0	✗	✗	0	67.5 ± 0.58
Rank-1	Linear	✗	✗	0	$\mathbf{67.8} \pm \mathbf{0.51}$
Rank-1	Linear	✗	✓	0	$\mathbf{67.6} \pm \mathbf{0.33}$

Experiments: Accuracy I

lossless on all fine-tuning tasks and comparable on pretraining tasks

4-bit AdamW:
- 1^{st}:B128/DE
- 2^{nd}:Rank-1/Linear

4-bit Factor:
- 1^{st}:B128/DE
- 2^{nd}:factorized

Table 2: Performance on language and vision tasks. Metric: NLU=Mean Accuracy/Correlation. CLS=Accuracy. NLG=BLEU. QA=F1. MT=SacreBleu. †: do not quantize optimizer states for embedding layers; ‡: $\beta_1 = 0$. See more results in App. A.

Optimizer	NLU RoBERTa-L	CLS Swin-T	NLG GPT-2 M	QA RoBERTa-L	MT Transformer
32-bit AdamW	88.9 ± 0.01	81.2 ± 0.05	67.7 ± 0.67	94.6 ± 0.13	26.61 ± 0.08
32-bit Adafactor	89.1 ± 0.00	80.0 ± 0.03	67.2 ± 0.81	94.6 ± 0.14	26.52 ± 0.02
32-bit Adafactor‡	89.3 ± 0.00	79.5 ± 0.05	67.2 ± 0.63	94.7 ± 0.10	26.45 ± 0.16
32-bit SM3	87.5 ± 0.00	79.0 ± 0.03	66.9 ± 0.58	91.7 ± 0.29	22.72 ± 0.09
8-bit AdamW†	89.1 ± 0.00	81.0 ± 0.01	67.5 ± 0.87	94.5 ± 0.04	26.66 ± 0.10
4-bit AdamW (ours)	89.1 ± 0.01	80.8 ± 0.02	67.8 ± 0.51	94.5 ± 0.10	26.28 ± 0.05
4-bit Factor (ours)	88.9 ± 0.00	80.9 ± 0.06	67.6 ± 0.33	94.6 ± 0.20	26.45 ± 0.05

Experiments: Accuracy II

performant on instruction fine-tuning tasks across model sizes

Table 3: Performance on LLaMA fine-tuning on MMLU and commonsense reasoning tasks across different sizes.

Model	Optimizer	MMLU (5-shot)	HellaSwag	ARC-e	ARC-c	OBQA
LLaMA-7B	Original	33.1	73.0	52.4	40.9	42.4
	32-bit AdamW	38.7	74.6	61.5	45.1	43.4
	4-bit AdamW	38.9	74.7	61.2	44.4	43.0
LLaMA-13B	Original	47.4	76.2	59.8	44.5	42.0
	32-bit AdamW	46.5	78.8	63.6	48.3	45.2
	4-bit AdamW	47.4	79.0	64.1	48.0	45.2
LLaMA-33B	Original	54.9	79.3	58.9	45.1	42.2
	32-bit AdamW	56.4	79.2	62.6	47.1	43.8
	4-bit AdamW	54.9	79.2	61.6	46.6	45.4

Experiments: Efficiency

Memory and Time with different optimizers

Table 4: Memory and Time of 4-bit optimizers compared with 32-bit AdamW and 8-bit Adam [15].

Task	Optimizer	Time	Total Mem.	Saved Mem.
LLaMA-7B	32-bit AdamW	3.35 h	75.40 GB	0.00 GB (0%)
	4-bit AdamW	3.07 h	31.87 GB	43.54 GB (57.7%)
	4-bit AdamW (fused)	3.11 h	31.88 GB	43.53 GB (57.7%)
RoBERTa-L	32-bit AdamW	3.93 min	5.31 GB	0.00 GB (0%)
	8-bit AdamW	3.38 min	3.34 GB	1.97 GB (37.1%)
	4-bit AdamW	5.59 min	3.02 GB	2.29 GB (43.1%)
	4-bit AdamW (fused)	3.17 min	3.00 GB	2.31 GB (43.5%)
	4-bit Factor	4.97 min	2.83 GB	2.48 GB (46.7%)
GPT-2 Medium	32-bit AdamW	2.13 h	6.89 GB	0.00 GB (0%)
	8-bit AdamW	2.04 h	4.92 GB	1.97 GB (28.6%)
	4-bit AdamW	2.43 h	4.62 GB	2.37 GB (34.4%)
	4-bit AdamW (fused)	2.11 h	4.62 GB	2.37 GB (34.4%)
	4-bit Factor	2.30 h	4.44 GB	2.45 GB (35.6%)

Summary

- We propose 4-bit AdamW and 4-bit Factor with quantization and factorization
- We evaluate our 4-bit optimizers on a wide range of tasks to showcase the effectiveness and efficiency
- Code released at: https://github.com/thu-ml/low-bit-optimizers