Towards Next-Level Post-Training Quantization of Hyper-Scale Transformers

Junhan Kim[*] **Chungman Lee**[*] **Eulrang Cho,** **Kyungphil Park,**
Ho-young Kim, **Joonyoung Kim,** **Yongkweon Jeon**[†]
Samsung Research
{jun_one.kim, chungman.lee, dragwon.jeon}@samsung.com

Abstract

With the increasing complexity of generative AI models, post-training quantization (PTQ) has emerged as a promising solution for deploying hyper-scale models on edge devices such as mobile and TVs. Existing PTQ schemes, however, consume considerable time and resources, which could be a bottleneck in real situations where frequent model updates and multiple hyperparameter tunings are required. As a cost-effective alternative, learning-free PTQ schemes have been proposed. However, the performance is somewhat limited because they cannot consider the inter-layer dependency within the attention module, which is a significant feature of Transformers. In this paper, we thus propose a novel PTQ algorithm that balances accuracy and efficiency. The key idea of the proposed algorithm called *aespa* is to perform quantization layer-wise for efficiency while targeting attention-wise reconstruction to consider the cross-layer dependency. Through extensive experiments on various language models and complexity analysis, we demonstrate that *aespa* is accurate and efficient in quantizing Transformer models. The code will be available at `https://github.com/SamsungLabs/aespa`.

1 Introduction

Model size has been gradually growing, resulting in deep generative models such as diffusion [25] and large-scale language models (LLMs) [29, 35] becoming more mainstream; the trend of AI is transitioning from discriminative models to generative models with numerous parameters in trillions. With the explosive growth in model complexity (parameters), the performance of AI models has been advancing and is now approaching or even exceeding human intelligence levels. However, this growth in scale has resulted in a corresponding increase in computational costs, which necessitates the efficient processing and compression of AI models. Interestingly, one attempts to expand the complexity of AI models to scale up performance, whereas the other aims to compress models to reduce cost.

Quantization is a promising solution and indispensable procedure facilitating the efficient deployment of AI models on devices that mainly support fixed-point arithmetic. By reducing the precision of weights, the memory bandwidth requirements can be relieved, and the embarrassing parallelism of quantized models can be SIMDified using highly efficient vector processing units such as NPU. To minimize the inevitable performance degradation caused by quantization, we can choose one of two approaches: quantization-aware training (QAT) [5, 14] and post-training quantization (PTQ) [23, 18]. Considering the model complexity and required resources (*e.g.,* training costs and available datasets), QAT is not practical for compressing models with billions of parameters. Consequently, recent quantization studies on hyper-scale Transformer [31] models have focused more on PTQ.

[*] Equal Contribution, [†] Corresponding Author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Although existing PTQ schemes have successfully quantized relatively small-scale models (*e.g.,* ResNet) [23, 10, 18, 6, 11], they have difficulty handling large-scale models because of their time and space complexity. As a cost-effective alternative, learning-free algorithms have been proposed recently [7, 13, 19], but their performance is somewhat limited because they do not consider the inter-layer dependency and are reliant on the nearest rounding. There is an accuracy-efficiency trade-off; thus, we aim to bridge the gap toward next-level quantization of hyper-scale Transformer models.

In this paper, we propose a novel PTQ algorithm, called *aespa*,[2] that pursues both accuracy and efficiency. The key idea of *aespa* is to perform quantization layer-wise for efficiency while targeting the attention-wise reconstruction to consider the cross-layer dependency.

Our contributions are summarized as follows:

- We propose a new quantization strategy that balances accuracy and efficiency. Our scheme aims to reconstruct the attention output to consider the cross-layer dependency while quantizing models layer-wise to pursue efficiency.
- To accelerate the quantization process, we propose refined quantization objectives for the attention module. Through a complexity analysis, we demonstrate that quantization that is approximately 10 times faster than existing block-wise approaches can be achieved by exploiting the proposed objectives.
- From extensive experiments on language models, we demonstrate that our approach outperforms conventional schemes by a significant margin, particularly for low-bit precision (INT2).

2 Background

2.1 Classic PTQ methods

Recent studies on PTQ have mostly attempted to minimize the increase in the task loss incurred by quantization rather than the quantization error itself ($\Delta \boldsymbol{W}$). Consider a pre-trained neural network parameterized by weights $\boldsymbol{W}$. If we assume the well-convergence of the network, the problem of quantizing weights $\boldsymbol{W}$ to minimize the loss degradation can be formulated as [16, 23]

$$\min_{\Delta \boldsymbol{w}} \ \mathbb{E}\left[\Delta \boldsymbol{w}^T \cdot \mathbf{H}^{(\boldsymbol{w})} \cdot \Delta \boldsymbol{w}\right], \tag{1}$$

where $\mathbf{H}^{(\boldsymbol{w})}$ is the Hessian related to the flattened weight $\boldsymbol{w}$. Because computing and storing $\mathbf{H}^{(\boldsymbol{w})}$ is infeasible, further assumptions have been made to simplify (1). In [23], for example, layer-wise independence has been assumed, relaxing (1) into the layer-wise reconstruction problem:

$$\min_{\Delta \boldsymbol{W}^{(\ell)}} \ \mathbb{E}\left[\left\| \mathcal{Q}(\boldsymbol{W}^{(\ell)})\boldsymbol{X} - \boldsymbol{W}^{(\ell)}\boldsymbol{X} \right\|_F^2\right], \tag{2}$$

where $\boldsymbol{W}^{(\ell)}$ denotes the weights of the ℓ-th layer, $\boldsymbol{X}$ is the input, and $\mathcal{Q}$ is a quantization function. For a uniform quantization, if the nearest-rounding is used to assign integer weights, $\mathcal{Q}$ is defined as

$$\mathcal{Q}(x) = s\left(\text{clamp}\left(\left\lfloor \frac{x}{s} \right\rceil + z, 0, 2^n - 1\right) - z\right), \tag{3}$$

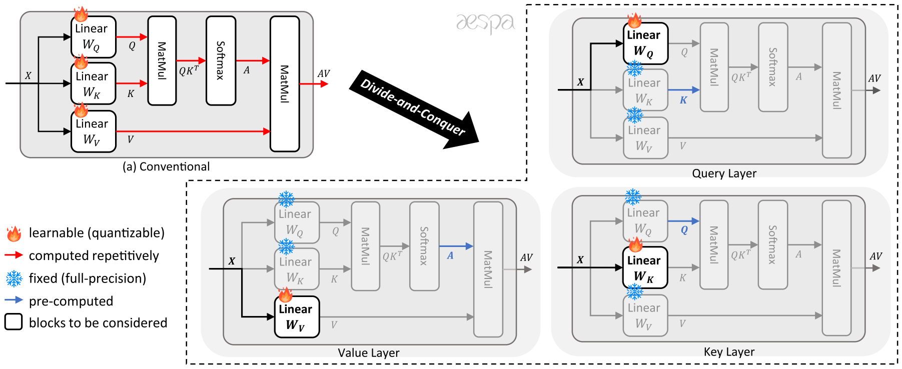
where s, z, and n are the scale, zero-point, and bit-width, respectively, and $\lfloor \cdot \rceil$ represents the round-off.

Early studies on PTQ focused on optimizing the weight-rounding policy [23, 10, 18, 11, 12]. These studies have attempted to assign each weight to a "proper" grid (instead of an adjacent grid), such that the loss degradation could be minimized. In [23], a learning-based weight-rounding optimization algorithm, called AdaRound, has been proposed to solve the layer-wise reconstruction problem in (2). In [18], AdaRound has been extended to the following block-wise reconstruction problem:

$$\min_{\Delta \boldsymbol{W}^{(\ell)}} \ \mathbb{E}\left[\left\| f^{(\ell)}\left(\mathcal{Q}(\boldsymbol{W}^{(\ell)}), \boldsymbol{X}\right) - f^{(\ell)}\left(\boldsymbol{W}^{(\ell)}, \boldsymbol{X}\right) \right\|_F^2\right], \tag{4}$$

where $\boldsymbol{W}^{(\ell)}$ denotes the weights of the ℓ-th block $f^{(\ell)}$ (*e.g.,* ResNet or Transformer block). By considering the dependency between layers inside the block, this algorithm, termed BRECQ, not only performs better than AdaRound, but also exhibits robust performance for a low bit-width (*e.g.,* INT2).

[2] *aespa*: attention-centric efficient and scalable post-training quantization algorithm



(a) Conventional

(b) Proposed

Figure 1: Overview of *aespa*. Each weight is quantized separately to reconstruct the attention output.

2.2 PTQ for LLMs

Although AdaRound and BRECQ have been successful in quantizing small-scale networks (*e.g.,* ResNet), scaling those learning-based schemes to LLMs with billions of parameters is challenging. In fact, BRECQ requires more than 20 GPU hours to quantize relatively small-sized language models (*e.g.,* OPT-2.7B; see Appendix K), which would not be suitable for the real-world deployment of LLMs where models to be deployed are frequently updated.

Owing to the excessive time and memory costs of classic PTQ schemes, recent studies have focused on developing cost-effective alternatives for quantizing LLMs. In OPTQ [7], a one-shot PTQ scheme that optimizes a weight-rounding policy without relying on learning, has been proposed. In addition, PTQ schemes that enhance the performance of the nearest-rounding, rather than optimizing the weight-rounding policy, have been proposed. These schemes use additional "foldable" parameters[3] to suppress activation outliers or quantize weights more precisely [33, 19, 13, 27, 20].

Although previous studies have mitigated the computational overhead of classic PTQ methods, they often sacrifice the low-bit quantization performance or suffer from an unstable quantization process. The main reason for this unsatisfactory performance is that all the schemes mentioned above, except OPTQ, rely on nearest-rounding and do not optimize the weight-rounding policy. Moreover, most of them target layer-wise reconstruction in (2), not block-wise reconstruction in (4), thus ignoring the cross-layer dependency within the attention module. Although [27, 20] target block-wise reconstruction via learning, they need to approximate gradients for a non-differentiable quantization function, which results in an unstable training process (see Table 1 in Section 4) [19].

Thus, we propose a novel PTQ scheme that balances accuracy and efficiency. In contrast to conventional LLM quantization methods, our scheme optimizes a weight-rounding policy while targeting block-wise reconstruction to consider the cross-layer dependency. The key difference over classic block-wise weight-rounding optimization is that we quantize models layer-wise for scalability, whereas layers are jointly quantized in the existing methods. Furthermore, we present an efficient pre-computation-based method for the computation of the block-wise objective in (4), which significantly reduces the computational overhead caused by repeated attention operations.

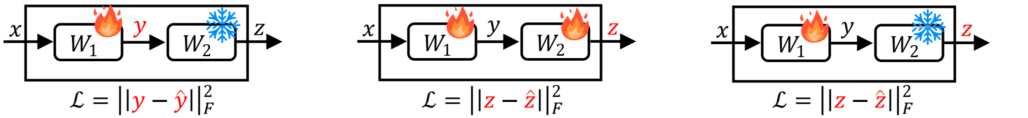
3 Method

3.1 Motivation

To gain insight into our approach, we first consider the objective of the layer-wise reconstruction in (2). Let $\Delta \boldsymbol{W}^{(\ell)} = \mathcal{Q}(\boldsymbol{W}^{(\ell)}) - \boldsymbol{W}^{(\ell)}$, then the reconstruction error can be expressed as

$$\mathbb{E}\left[\|\Delta \boldsymbol{W}\boldsymbol{X}\|_F^2\right] = \mathbb{E}\left[\operatorname{tr}\left(\Delta \boldsymbol{W}\boldsymbol{X}\boldsymbol{X}^T\Delta \boldsymbol{W}^T\right)\right] = \operatorname{tr}\left(\Delta \boldsymbol{W}\cdot\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right]\cdot\Delta \boldsymbol{W}^T\right). \tag{5}$$

[3] By foldable parameters, we mean the parameters that can be merged into other layers within the Transformer block (*e.g.,* LayerNorm), thereby imposing no extra computational cost during the inference [13].


x
W_1
y
W_2
z
$\mathcal{L} = ||y - \hat{y}||_F^2$
x
W_1
y
W_2
z
$\mathcal{L} = ||z - \hat{z}||_F^2$
x
W_1
y
W_2
z
$\mathcal{L} = ||z - \hat{z}||_F^2$

(a) **Layer**-Wise, **Layer** Output (b) **Block**-Wise, **Block** Output (c) **Layer**-Wise, **Block** Output (Proposed)

Figure 2: Quantization strategies (simplified)

Consequently, the layer-wise quantization problem can be recast as follows:

$$\min_{\Delta \boldsymbol{W}} \operatorname{tr}\left(\Delta \boldsymbol{W} \cdot \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right] \cdot \Delta \boldsymbol{W}^T\right). \tag{6}$$

The new form of the quantization objective in (6) implies that if $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ is pre-computed and stored before quantization, we can measure the reconstruction error over the entire calibration dataset with a single matrix multiplication and element-wise multiplication.[4] This is in contrast to the original formulation in (2) which requires the computation of $\mathcal{Q}(\boldsymbol{W})\boldsymbol{X}$ or $\Delta\boldsymbol{W}\boldsymbol{X}$ for every input $\boldsymbol{X}$.

A natural question that arises from this finding is *"Can we also measure the block reconstruction error efficiently based on such a pre-computation?"*. In the following subsections, we describe our main strategy to simplify block-wise quantization and then present a refined objective for the attention module, where the objective can be computed efficiently with certain pre-computed values.

3.2 Quantization strategy of *aespa*

When quantizing the attention module using conventional block-wise reconstruction methods (Figure 1(a)), the query, key, and value projections have been jointly optimized such that

$$\min_{\Delta \boldsymbol{W}_Q, \Delta \boldsymbol{W}_K, \Delta \boldsymbol{W}_V} \mathbb{E}\left[\left\|\operatorname{SA}(\widehat{\boldsymbol{Q}}, \widehat{\boldsymbol{K}}, \widehat{\boldsymbol{V}}) - \operatorname{SA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})\right\|_F^2\right], \tag{7}$$

where the output of attention module $\operatorname{SA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is defined as

$$\operatorname{SA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \operatorname{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V} = \boldsymbol{A}\boldsymbol{V}. \tag{8}$$

In such a case, we need to compute $\operatorname{SA}(\widehat{\boldsymbol{Q}}, \widehat{\boldsymbol{K}}, \widehat{\boldsymbol{V}})$ for every batch sequence in each iteration, which is computationally heavy and time-consuming (see Section 3.5 for details on complexity).

To overcome this computational overhead, we quantize each projection *separately* in a divide-and-conquer manner. For example, when quantizing the query projection $\boldsymbol{W}_Q$, we fix $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ with full-precision (Figure 1(b)), which facilitates the factoring out of common terms affected by $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ (see Section 3.3 for details). We emphasize that this strategy differs from conventional layer-wise quantization schemes (*e.g.,* AdaRound and OPTQ) in that we aim to minimize the reconstruction error for the attention module, not the reconstruction error for each layer.

We conduct experiments to demonstrate the importance of targeting attention-wise reconstruction and validity of the proposed quantization strategy. In our experiments, we set the loss function for each projection as the attention reconstruction error in (7) but quantize each projection separately (see Figure 2(c)). Table 5 in Appendix B summarizes the performance of AdaRound, BRECQ, and our approach. As evident, our approach uniformly outperforms AdaRound for all bit-widths, although both methods quantize models layer-wise. This is because we can consider cross-layer dependency (*i.e.,* relationship between the query, key, and value) by targeting attention-wise reconstruction, which is different from AdaRound wherein layers are considered independent. Furthermore, once we target attention-wise reconstruction, separate layer-wise quantization does not incur severe performance degradation compared to the joint quantization method (BRECQ). In fact, our approach causes only a marginal performance degradation for 2-bit and exhibits comparable performance for 3-bit and 4-bit. For further discussion on the proposed strategy, see Appendix B.

3.3 Refined quantization objectives for *aespa*

One might ask whether our strategy incurs more computational cost than that required by the joint quantization because we update only one layer at a time (see Figure 1(b)). This is in contrast

[4] We note that the computation of $\operatorname{tr}(\mathbf{A}\mathbf{B}\mathbf{C}^T)$ can be implemented as `torch.sum((`$\mathbf{A}\mathbf{B}$`)` $\odot$ $\mathbf{C}$`)`, where $\odot$ denotes the element-wise product operation. They are mathematically equivalent.

to existing methods, in which the layers inside the attention module are updated simultaneously (Figure 1(a)). To reduce this additional cost, we refine the quantization objective in (7) for each projection.

Value projection When quantizing the value projection $\boldsymbol{W}_V$, the query and key projections are fixed with full-precision. In this case, by factoring out the common term influenced by $\boldsymbol{Q}$ and $\boldsymbol{K}$, we can simplify the attention reconstruction error $\Delta \mathrm{SA}_V$ as follows:

$$\Delta \mathrm{SA}_V = \mathbb{E}\left[\left\|\boldsymbol{A}\widehat{\boldsymbol{V}} - \boldsymbol{A}\boldsymbol{V}\right\|_F^2\right] = \mathbb{E}\left[\left\|\boldsymbol{A}\Delta\boldsymbol{V}\right\|_F^2\right] = \mathbb{E}\left[\left\|\Delta\boldsymbol{W}_V\boldsymbol{X}\boldsymbol{A}^T\right\|_F^2\right]. \tag{9}$$

Thus, the problem to quantize $\boldsymbol{W}_V$ to minimize the attention reconstruction error can be recast as

$$\min_{\Delta\boldsymbol{W}_V} \mathbb{E}\left[\left\|\Delta\boldsymbol{W}_V\boldsymbol{X}\boldsymbol{A}^T\right\|_F^2\right]. \tag{10}$$

Query projection When the key and value projections are fixed with full-precision, the attention reconstruction error $\Delta \mathrm{SA}_Q$ caused by $\Delta\boldsymbol{W}_Q$ is expressed as

$$\Delta \mathrm{SA}_Q = \mathbb{E}\left[\left\|\mathrm{SA}(\widehat{\boldsymbol{Q}}, \boldsymbol{K}, \boldsymbol{V}) - \mathrm{SA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})\right\|_F^2\right] = \mathbb{E}\left[\left\|\Delta\boldsymbol{A}\boldsymbol{V}\right\|_F^2\right], \tag{11}$$

where $\Delta\boldsymbol{A} = \mathrm{softmax}(\widehat{\boldsymbol{Q}}\boldsymbol{K}^T/\sqrt{d}) - \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d})$. To avoid the computational overhead of repetitive softmax operations, we approximate $\Delta\boldsymbol{A}$ with its first-order Taylor series as

$$\Delta\boldsymbol{A} \approx \frac{\Delta\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}} \cdot \mathbf{J}_{\mathrm{softmax}}^T, \tag{12}$$

where $\mathbf{J}_{\mathrm{softmax}}$ is the Jacobian of the softmax function. By combining (11) and (12), we obtain

$$\Delta \mathrm{SA}_Q \approx \frac{1}{d}\mathbb{E}\left[\left\|\Delta\boldsymbol{Q}\boldsymbol{K}^T\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\right\|_F^2\right] = \frac{1}{d}\mathbb{E}\left[\left\|\boldsymbol{V}^T\mathbf{J}_{\mathrm{softmax}}\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2\right]. \tag{13}$$

Although we can circumvent conducting attention operations using the modified form in (13), a large amount of memory is required to store the Jacobian $\mathbf{J}_{\mathrm{softmax}}$ (*e.g.,* more than 100 GB of memory for OPT-125M).[5] As a cost-effective alternative, we build an upper bound of (13) and then employ it as a surrogate of $\Delta \mathrm{SA}_Q$ when quantizing $\boldsymbol{W}_Q$. Specifically, by noting that

$$\left\|\boldsymbol{V}^T\mathbf{J}_{\mathrm{softmax}}\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2 \leq \left\|\boldsymbol{V}^T\mathbf{J}_{\mathrm{softmax}}\right\|_F^2 \cdot \left\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2 \tag{14}$$

and the term $\|\boldsymbol{V}^T\mathbf{J}_{\mathrm{softmax}}\|_F^2$ is fixed in the quantization process, we minimize $\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\|_F^2$ with the hope that $\Delta \mathrm{SA}_Q$ also decreases. In other words, our quantization objective for $\boldsymbol{W}_Q$ is

$$\min_{\Delta\boldsymbol{W}_Q} \mathbb{E}\left[\left\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2\right]. \tag{15}$$

Key projection By taking similar steps, the quantization objective for the key projection $\boldsymbol{W}_K$ can be formulated as (see Appendix C for the detailed derivation)

$$\min_{\Delta\boldsymbol{W}_K} \mathbb{E}\left[\left\|\boldsymbol{Q}\Delta\boldsymbol{W}_K\boldsymbol{X}\right\|_F^2\right]. \tag{16}$$

3.4 Algorithm description

The proposed *aespa* consists of two main steps. Specifically, *aespa* first determines the quantization parameters (*i.e.,* scale and zero-point) and then optimizes an integer weight $\boldsymbol{W}_{int}$ for each weight.

Note that we only used the definition of the attention operation when developing the refined objectives in (10), (15), and (16). Thus, our objectives can be integrated into any layer-wise quantization scheme without effort. For example, we can compute the quantization parameters by combining existing parameter initialization algorithms (*e.g.,* AWQ [19] and Z-FOLD [13]) with the proposed objectives. We can also optimize a weight-rounding policy using conventional methods (*e.g.,* AdaRound [23])

[5] Note that the shape of $\mathbf{J}_{\mathrm{softmax}}$ is $[L, L, L]$ (L is the input sequence length) for each attention head because $\mathbf{J}_{\mathrm{softmax}}(\boldsymbol{a}_\ell) = \mathrm{diag}(\boldsymbol{a}_\ell) - \boldsymbol{a}_\ell^T\boldsymbol{a}_\ell \in \mathbb{R}^{L\times L}$ for each row $\boldsymbol{a}_\ell$ of $\boldsymbol{A}$.

together with our objectives (see Appendix F for details). In the proposed *aespa*, we use Z-FOLD in computing the quantization parameters and employ AdaRound in optimizing a weight-rounding policy. In Algorithm 1 (see Appendix A), we summarize the proposed *aespa*.[6]

To accelerate the weight-rounding learning process, we further modify the objective functions such that the value can be computed efficiently via pre-computation, as in (5).

Modified objective for (10) The proposed objective for the value projection can be recast as

$$\mathbb{E}\left[\left\|\Delta \boldsymbol{W}_V \boldsymbol{X} \boldsymbol{A}^T\right\|_F^2\right] = \operatorname{tr}\left(\Delta \boldsymbol{W}_V \mathbb{E}\left[\boldsymbol{X} \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{X}^T\right] \Delta \boldsymbol{W}_V^T\right). \tag{17}$$

The modified objective allows us to perform each iteration of the weight-rounding learning efficiently. Specifically, by computing $\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ before quantization and reusing it in the quantization process[7], we can avoid the overhead of computing $\left\|\Delta \boldsymbol{W}_V \boldsymbol{X} \boldsymbol{A}^T\right\|_F^2$ for every input $\boldsymbol{X}$ and compute the loss with one simple matrix multiplication and a single element-wise multiplication (see Footnote 4).

Another intriguing feature of this modification is that it facilitates a more reliable update of $\Delta \boldsymbol{W}_V$ than the original objective in (10). Specifically, because $\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ is pre-computed using all calibration data, the loss computed with (17) considers the entire calibration dataset (*i.e.,* the batch size is the total number of data). Thus, a better estimate of the true gradient can be obtained without any memory issues, which could lead to more consistent updates of $\Delta \boldsymbol{W}_V$ and faster convergence [28].

The modified objective in (17) also implies that the Hessian $\mathbf{H}_V$ for each row of $\boldsymbol{W}_V$ is

$$\mathbf{H}_V = 2\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]. \tag{18}$$

We note that the proposed Hessian $\mathbf{H}_V$ differs from $\mathbf{H} = 2\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$, which has been commonly used as an approximated Hessian in conventional methods [6, 7, 13, 3]. The key reason for the difference is that we consider the dependency between $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ by targeting attention-wise reconstruction, whereas the previous methods assumed independence. To observe the effect of considering the cross-layer dependency, we use different Hessians (*i.e.,* $\mathbf{H}_V$ and $\mathbf{H}$) when quantizing language models and then compare the performance of the quantized models (see Appendix D). Evidently, the quantization performance is much better when the proposed Hessian $\mathbf{H}_V$ is used, which demonstrates the importance of considering the cross-layer dependency.

Modified objectives for (15) and (16) If we denote the vectorized representation of $\Delta \boldsymbol{W}_Q$ as $\Delta \boldsymbol{w}_Q$, the proposed objective in (15) can be expressed as (see Appendix E for the derivation)

$$\mathbb{E}\left[\|\boldsymbol{K}\Delta \boldsymbol{W}_Q \boldsymbol{X}\|_F^2\right] = \Delta \boldsymbol{w}_Q^T \cdot \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T \otimes \boldsymbol{K}^T\boldsymbol{K}\right] \cdot \Delta \boldsymbol{w}_Q. \tag{19}$$

where $\otimes$ is the Kronecker product operation. To reduce the memory cost of storing the Kronecker product term $\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T \otimes \boldsymbol{K}^T\boldsymbol{K}\right]$, we approximate it as [2]

$$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T \otimes \boldsymbol{K}^T\boldsymbol{K}\right] \approx \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right] \otimes \mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K}\right]. \tag{20}$$

By combining (19) and (20), we obtain

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{K}\Delta \boldsymbol{W}_Q \boldsymbol{X}\|_F^2\right] &\approx \Delta \boldsymbol{w}_Q^T \cdot \left(\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right] \otimes \mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K}\right]\right) \cdot \Delta \boldsymbol{w}_Q \\ &\overset{(a)}{=} \operatorname{tr}\left(\mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K}\right] \Delta \boldsymbol{W}_Q \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right] \Delta \boldsymbol{W}_Q^T\right),\end{aligned} \tag{21}$$

where the proof of (a) is provided in Appendix E. By taking similar steps, the objective for the key projection can be recast as

$$\mathbb{E}\left[\|\boldsymbol{Q}\Delta \boldsymbol{W}_K \boldsymbol{X}\|_F^2\right] = \operatorname{tr}\left(\mathbb{E}\left[\boldsymbol{Q}^T\boldsymbol{Q}\right] \Delta \boldsymbol{W}_K \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right] \Delta \boldsymbol{W}_K^T\right). \tag{22}$$

The modified objectives in (21) and (22) imply that the loss over the total calibration dataset can be calculated efficiently by computing $\mathbb{E}[\boldsymbol{K}^T\boldsymbol{K}]$, $\mathbb{E}[\boldsymbol{Q}^T\boldsymbol{Q}]$, and $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ in advance.

[6] We use the layer-wise objective in (6) for the weights other than the query, key, and value projections (*i.e.,* out-projection and weights inside the feed-forward network).

[7] The term $\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ is not affected by $\Delta \boldsymbol{W}_V$ and thus fixed in the quantization process.

3.5 Complexity analysis for *aespa*

We discuss the computational complexity of *aespa*. Specifically, we analyze the number of floating-point operations (flops) required to perform one iteration for weight-rounding optimization (line 6 in Algorithm 1). For each projection, the required number of flops is summarized as follows.

- **Value**: By reusing the pre-computed $\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$, the loss value in (17) can be computed with one matrix multiplication and one element-wise multiplication/addition (see Footnote 4). The associated cost is $2d_hd^2 + d_hd - 1$ flops, where d is the hidden size and d_h is the input dimension for each attention head.
- **Query/key**: Once $\mathbb{E}[\boldsymbol{K}^T\boldsymbol{K}]$, $\mathbb{E}[\boldsymbol{Q}^T\boldsymbol{Q}]$, and $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ have been computed in advance, the loss values in (21) and (22) can be computed by performing two matrix multiplications and one element-wise multiplication/addition. This requires $2d_hd^2 + 2d_h^2d - 1$ flops for each projection.

To summarize, the total number of flops required in each iteration of the proposed *aespa* is

$$\mathcal{C}_{aespa} = 6d_hd^2 + 4d_h^2d + d_hd - 3 = \mathcal{O}(d_hd^2). \tag{23}$$

We emphasize that regardless of the amount of calibration data, the number of flops to compute the loss considering the entire dataset is fixed as $\mathcal{C}_{aespa}$.

We now compare the complexities of *aespa* and conventional block-wise quantization methods. It can be easily verified that the existing methods require the following number of flops for handling B input sequences of length L (see Appendix G):

$$\mathcal{C}_{exist} = B(6d_hdL + 4d_hL^2 + 2L^2 - L - 1) = \mathcal{O}(Bd_hL \cdot \max\{d, L\}). \tag{24}$$

Table 7 in Appendix G summarizes the computational costs for different sizes of OPT models. For the conventional methods, we report the cost of using four sequences in each iteration ($B = 4$). We observe that the computational cost of *aespa* is considerably lower than that of conventional methods. In particular, for small-scale models, *aespa* performs ten times fewer number of flops. It can be observed that the gap between $\mathcal{C}_{aespa}$ and $\mathcal{C}_{exist}$ decreases as the model size increases. This is because the hidden size d exceeds the sequence length L (which is fixed for all models) for large models. Nevertheless, *aespa* still incurs a lower computational cost, and the gap increases if conventional methods use larger batch sizes.

4 Experimental results

4.1 Experimental setup

We quantize publicly available LLMs (*e.g.,* OPT [35], BLOOM [26], LLaMA [29], and LLaMA2 [30]) using the proposed *aespa*. When implementing *aespa*, we compute the quantization parameters with Z-FOLD [13] and optimize a weight-rounding policy via AdaRound [23], where the proposed row-wise Hessians and loss functions (see Table 4 in Appendix A) are utilized instead of the existing ones. When computing the quantization parameters, we follow the stopping criterion introduced by [13]. Before optimizing a weight-rounding policy, we update the full-precision weights via OPTQ [7], which empirically reduces the number of iterations required for weight-rounding optimization. When optimizing a weight-rounding policy, we set the number of iterations, learning rate, and weight of the rounding loss (see λ in (28)) to 2,000, 0.015, and 1.5, respectively.

When constructing the calibration dataset, we randomly sample 128 segments consisting of 2048 tokens from the C4 dataset [24] as in [7, 13, 3]. In our experiments, we quantize only weights and retain activations in full-precision because activations are not a significant bottleneck for LLMs [7] and the inference of LLMs can be accelerated sufficiently by reducing memory movement through weight quantization [15]. We evaluate the performance of the quantized models using benchmark datasets (*e.g.,* WikiText-2 [22], C4 [24], and PTB [21]) and zero-shot tasks. Except for the experiments on the LLaMA2 models, which were performed using an NVIDIA H100 GPU, we conducted all experiments using a single NVIDIA A100 GPU (80 GB).

4.2 Comparison with prior arts

Comparison with block-wise PTQ schemes We compare the proposed *aespa* with conventional block-wise PTQ methods, among which BRECQ is a classic weight-rounding optimization method,

Table 1: Performance (PPL ↓) of the proposed *aespa* and conventional block-wise PTQ methods.

(a) WikiText-2

Precision	Method	OPT				LLaMA			LLaMA2	
		125M	1.3B	2.7B	6.7B	7B	13B	30B	7B	13B
FP16	Baseline	27.65	14.63	12.47	10.86	5.677	5.091	4.101	5.472	4.884
INT3	Brecq [18]	33.25	16.09	13.37	OOM	OOM	OOM	OOM	OOM	OOM
	OmniQuant [27]	39.14	17.59	14.87	12.87	6.716	5.798	4.963	6.798	5.751
	AffineQuant [20]	36.15	17.26	14.25	12.30	6.712	5.820	4.951	6.795	5.757
	aespa	**32.71**	**15.79**	**13.14**	**11.23**	**6.579**	**5.611**	**4.688**	**6.241**	**5.462**
INT2	Brecq [18]	**60.38**	56.25	113.6	OOM	OOM	OOM	OOM	OOM	OOM
	OmniQuant [27]	NaN	399.6	1.6e3	4.9e3	18.18	NaN	10.15	35.40	20.19
	AffineQuant [20]	143.9	56.45	35.16	25.32	18.83	11.08	NaN	NaN	18.49
	aespa	71.18	**24.26**	**22.22**	**15.71**	**11.94**	**10.30**	**7.845**	**13.99**	**12.14**

(b) C4

Precision	Method	OPT				LLaMA			LLaMA2	
		125M	1.3B	2.7B	6.7B	7B	13B	30B	7B	13B
FP16	Baseline	26.56	16.07	14.34	12.71	7.344	6.798	6.131	7.264	6.727
INT3	Brecq [18]	29.74	17.46	15.39	OOM	OOM	OOM	OOM	OOM	OOM
	OmniQuant [27]	34.92	18.83	16.80	14.21	8.605	7.604	6.822	9.085	7.821
	AffineQuant [20]	32.78	18.27	16.11	13.80	8.631	7.609	6.803	9.059	7.732
	aespa	**29.51**	**17.10**	**15.27**	**13.15**	**8.465**	**7.399**	**6.634**	**8.225**	**7.392**
INT2	Brecq [18]	**47.85**	41.05	83.32	OOM	OOM	OOM	OOM	OOM	OOM
	OmniQuant [27]	NaN	239.1	1.1e3	4.4e3	18.59	NaN	14.74	26.27	18.93
	AffineQuant [20]	95.86	43.66	29.75	24.04	16.87	12.79	NaN	NaN	15.20
	aespa	56.88	**23.54**	**22.53**	**17.28**	**13.63**	**11.46**	**10.35**	**14.36**	**13.59**

* ‘NaN’ means that loss diverges in the quantization process.

* ‘OOM’ means that out-of-memory issues occur when quantizing models with a single A100 GPU.

* Results for high bit-widths are provided in Appendix H due to the page limitation.

and OmniQuant and AffineQuant are LLM quantization methods that mitigate the computational overhead of Brecq by learning only a few quantization and foldable parameters [27, 20]. For OmniQuant and AffineQuant, we ran the official codes[8] provided by the authors. For both methods, we activated the learnable equivalent transformation (LET) and learnable weight clipping (LWC) options and reported the obtained results. When implementing Brecq, we employed the hyperparameter settings provided in [18]. In this comparison, the BLOOM models and OPT-350M were excluded because they are not supported by OmniQuant and AffineQuant.

As Table 1 shows, *aespa* uniformly outperforms OmniQuant/AffineQuant.[9] In particular, the performance gap is significant for 2-bit; while OmniQuant/AffineQuant suffer from instability (*i.e.,* loss diverges) or collapse (perplexity (PPL) $> 10^3$), *aespa* exhibits reasonable PPL. The outstanding performance is attributed to the fact that *aespa* optimizes a weight-rounding policy after determining the quantization parameters (lines 5-8 in Algorithm 1), whereas OmniQuant/AffineQuant rely on the naive nearest rounding and approximate gradients for the non-differentiable quantization function.

Although Brecq performs best for the 2-bit quantization of OPT-125M, it lacks scalability; Brecq requires approximately 20 GPU hours for a relatively small-scale OPT-2.7B (see Table 14 in Appendix K). Even for OPT-125M, Brecq requires approximately 2 GPU hours, whereas the proposed *aespa* completes quantization in 5 minutes. One might wonder why the performance of Brecq worsens as the model size increases. We assume that this is attributable to the choice of hyperparameters (*e.g.,* learning rate and weight of rounding loss). In fact, the hyperparameters presented in [18] have been optimized for ImageNet, but not for LLMs. It is expected that we can obtain better performance for Brecq via deliberate hyperparameter tuning; however, this would not be feasible for real-world deployment because it requires considerable time (see Table 14 in Appendix K).

Comparison with layer-wise PTQ schemes We compare the proposed *aespa* with conventional layer-wise PTQ schemes, among which RTN is the method that naively assigns the nearest grid, OPTQ is a backpropagation-free weight-rounding optimization algorithm [7], and Z-Fold is the

[8] https://github.com/OpenGVLab/OmniQuant, https://github.com/bytedance/AffineQuant

[9] We note that our results are different from those reported in [27, 20] where a different calibration dataset (WikiText-2) was used; see Appendix L for more discussion on this issue.

Table 2: Performance (PPL ↓) of *aespa* and existing layer-wise PTQ methods on BLOOM models.

Precision	Method	WikiText-2					C4				
		560M	1.1B	1.7B	3B	7.1B	560M	1.1B	1.7B	3B	7.1B
FP16	Baseline	22.42	17.69	15.39	13.48	11.37	26.60	22.05	19.49	17.49	15.20
INT3	RTN	56.74	49.85	63.37	39.07	17.35	66.99	60.41	113.6	79.84	22.54
	OPTQ	31.55	23.84	20.06	17.13	13.56	34.62	27.62	23.87	20.96	17.43
	Z-FOLD	26.52	20.99	17.39	15.11	12.26	29.97	24.43	21.52	19.01	16.12
	aespa	**25.39**	**19.81**	**16.95**	**14.68**	**12.00**	**29.10**	**23.80**	**20.93**	**18.55**	**15.91**
INT2	RTN	7.8e5	9.8e5	3.5e5	1.4e5	2.1e5	1.4e6	2.1e6	2.7e5	9.2e4	1.3e5
	OPTQ	1.7e3	1.9e3	1.4e3	796.5	194.2	533.4	538.0	562.9	351.6	112.8
	Z-FOLD	65.45	44.50	35.69	27.40	18.87	64.11	42.96	37.26	32.64	22.46
	aespa	**44.91**	**34.12**	**27.67**	**21.65**	**16.31**	**45.04**	**35.12**	**29.95**	**25.04**	**20.00**

* Results for high bit-widths and other language models (*e.g.,* OPT, LLaMA, and LLaMA2) are provided in Appendix I.

method exploiting additional foldable parameters to quantize weights more elaborately [13]. Table 2 and Tables 9-12 (see Appendix I) summarize the results for the OPT, BLOOM, LLaMA, and LLaMA2 models of various sizes. Evidently, *aespa* uniformly outperforms conventional schemes, regardless of the size and type of LLMs. In particular, for 2-bit, there exists a significant performance gap between *aespa* and existing methods; the PPL obtained by *aespa* is twice as low as those of conventional methods for small-scale models (*e.g.,* OPT-125M). The key factors leading to such an outstanding performance are: 1) the consideration of the cross-layer dependency achieved by targeting attention-wise reconstruction, and 2) efficient weight-rounding optimization based on pre-computations.

Zero-shot task performance We evaluate the reasoning performance of quantized models using zero-shot tasks (*e.g.,* ARC [4], HellaSwag [34], and MMLU [8]). We note that the zero-shot setting was ensured in our experiments because we used excerpts from randomly crawled websites (not task-specific data) as a calibration dataset. From the zero-shot results in Table 3 and Table 13 (see Appendix J), we observe that the proposed *aespa* performs the best in almost all cases, and the performance gap between *aespa* and the existing methods is large for 2-bit.

Time cost We summarize the processing times of the different quantization algorithms in Appendix K. We note that the processing time of *aespa* includes the time required for pre-computations (lines 2-4 in Algorithm 1). As expected, *aespa* completes quantization much faster than BRECQ. For example, while BRECQ requires more than 10 GPU hours for OPT-1.3B, *aespa* completes quantization in 1.24 hours, which demonstrates the effectiveness of the proposed pre-computation-based loss computation strategy. Although other block-wise methods (OmniQuant/AffineQuant) perform quantization faster than *aespa* for hyper-scale models, they suffer from unstable training processes or exhibit poor PPL performance (*e.g.,* PPL of OmniQuant is larger than 10^3 for OPT-6.7B; see Table 1). In addition, we observe that OPTQ performs quantization quickly, but its 2-bit performance collapses regardless of the model size (see Table 9 in Appendix I). Except for *aespa*, Z-FOLD is the only method that shows both reasonable performance and processing time.

Discussion In real situations, when one needs to preserve the performance of the original model as much as possible, the proposed *aespa* would be an intriguing solution. In particular, when deploying LLMs on resource-constrained platforms where up to 7B models are commonly employed (*e.g.,* mobile devices), *aespa* would be a good fit. Even when fast quantization of hyper-scale models is required, *aespa* can be used with a slight modification. Specifically, in time-limited cases, one can skip the weight-rounding optimization (lines 5-8 in Algorithm 1) and simply determine the quantization parameters using the proposed Hessian that considers the cross-layer dependency (line 4 in Algorithm 1). In doing so, we can not only save the time required to optimize a weight-rounding mechanism, but also save the memory required to store pre-computed values ($\mathbb{E}[\mathbf{K}^T\mathbf{K}]$ and $\mathbb{E}[\mathbf{Q}^T\mathbf{Q}]$). Indeed, when performing only quantization parameter computation, we achieved a significant reduction in the processing time (see Table 15 in Appendix K) while still exhibiting better performance than conventional methods (see Table 6 in Appendix D).

5 Conclusion

We proposed a next-level PTQ scheme for Transformers, called *aespa*. By targeting the attention-wise reconstruction while quantizing Transformers layer-wise, we could consider the cross-layer dependency within the attention module and complete the quantization much faster than the existing

Table 3: INT2 zero-shot performance (accuracy ↑) of *aespa* and existing methods.

Model	Method	ARC-c	ARC-e	HellaSwag	MMLU	Average
LLaMA-7B	FP16	44.62	72.85	76.18	32.19	56.46
	RTN	28.67	25.00	26.43	25.72	26.46
	OPTQ [7]	29.18	26.14	26.18	24.04	26.39
	Z-Fold [13]	30.63	52.44	53.55	23.27	39.97
	OmniQuant [27]	27.22	49.20	50.65	23.74	37.70
	AffineQuant [20]	27.90	49.58	51.85	24.15	38.37
	aespa	33.36	55.64	58.31	23.12	**42.61**
LLaMA-13B	FP16	47.87	74.75	79.08	43.46	61.29
	RTN	28.16	27.15	26.09	25.53	26.73
	OPTQ [7]	27.22	25.76	25.67	25.05	25.93
	Z-Fold [13]	32.68	58.08	57.89	26.44	43.77
	OmniQuant [27]	NaN	NaN	NaN	NaN	NaN
	AffineQuant [20]	32.17	56.36	60.29	25.22	43.51
	aespa	34.73	61.49	62.68	28.74	**46.91**
LLaMA-30B	FP16	52.90	78.96	82.63	54.66	67.29
	RTN	27.05	26.39	25.87	25.48	26.20
	OPTQ [7]	27.13	26.60	26.12	23.56	25.85
	Z-Fold [13]	39.93	65.07	65.89	30.85	50.44
	OmniQuant [27]	34.22	58.50	64.83	25.91	45.87
	AffineQuant [20]	NaN	NaN	NaN	NaN	NaN
	aespa	41.13	67.00	67.90	35.67	**52.93**
LLaMA2-7B	FP16	46.16	74.49	75.99	41.87	59.63
	RTN	28.33	26.01	25.88	23.02	25.81
	OPTQ [7]	26.37	26.09	25.11	25.10	25.67
	Z-Fold [13]	26.62	42.68	44.71	22.88	34.22
	OmniQuant [27]	25.00	38.80	42.97	23.03	32.45
	AffineQuant [20]	NaN	NaN	NaN	NaN	NaN
	aespa	30.29	51.47	56.75	25.59	**41.03**
LLaMA2-13B	FP16	49.06	77.44	79.39	52.10	64.50
	RTN	27.22	25.04	25.58	24.69	25.63
	OPTQ [7]	26.71	27.19	25.42	23.74	25.77
	Z-Fold [13]	28.41	48.32	51.59	23.98	38.08
	OmniQuant [27]	27.13	47.98	53.27	23.81	38.05
	AffineQuant [20]	30.80	52.90	57.74	24.45	41.47
	aespa	31.91	55.18	55.49	29.97	**43.14**

* 'NaN' means that loss diverges in the quantization process.

* Results for high bit-widths are provided in Appendix J due to the page limitation.

approach for block-wise reconstruction (*i.e.,* Brecq). Extensive experiments on language models have demonstrated the outstanding performance of *aespa*.

Limitations and future work While we focused on the attention output, the output of the entire Transformer block (containing fully connected layers) can be used to consider the dependencies between more layers. However, in this case, the objective functions would be more complicated than those in (13) and (25) due to nonlinear activation functions (*e.g.,* SiLU for LLaMA models), normalization layers, and weights of larger dimensions. Enhancing the quantization performance by developing an efficient form of the reconstruction error for the Transformer block would be an interesting future work. Furthermore, while we focused on weight-only quantization, activations may need to be quantized to deploy AI models on integer-only arithmetic hardware (*e.g.,* NPU). Extending the proposed *aespa* for weight-activation quantization by integrating existing techniques to suppress activation outliers [33, 1] is also an interesting research direction. Finally, while we verified the performance of *aespa* with LLMs, we believe that *aespa* can also be used for the quantization of diffusion models. To that end, we may need to incorporate some diffusion-specific quantization strategies to overcome output distribution discrepancies over different time steps (*e.g.,* grouping of time-steps with similar distributions [32], temporal feature preservation [9], and separate quantization for shortcuts in U-Net [17]), which will be considered in our future studies.

References

[1] 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. *arXiv:2404.00456*, 2024.

[2] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In *International Conference on Machine Learning*, pages 557–565. PMLR, 2017.

[3] 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.

[4] 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:1803.05457v1*, 2018.

[5] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2019.

[6] Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

[7] 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.

[8] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.

[9] Yushi Huang, Ruihao Gong, Jing Liu, Tianlong Chen, and Xianglong Liu. TFMQ-DM: Temporal feature maintenance quantization for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7362–7371, 2024.

[10] 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*, pages 4466–4475. PMLR, 2021.

[11] Yongkweon Jeon, Chungman Lee, Eulrang Cho, and Yeonju Ro. Mr. BiQ: Post-training non-uniform quantization based on minimizing the reconstruction error. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12329–12338, 2022.

[12] Yongkweon Jeon, Chungman Lee, and Ho-young Kim. GENIE: show me the data for quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12064–12073, 2023.

[13] Yongkweon Jeon, Chungman Lee, Kyungphil Park, and Ho-young Kim. A frustratingly easy post-training quantization scheme for llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14446–14461, 2023.

[14] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[15] 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.

[16] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS)*, volume 2, pages 598–605, 1989.

[17] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-Diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545, 2023.

[18] Yuhang 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 *International Conference on Learning Representations (ICLR)*, 2021.

[19] 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 llm compression and acceleration. In *MLSys*, 2024.

[20] Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. AffineQuant: Affine transformation quantization for large language models. *arXiv:2403.12544*, 2024.

[21] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

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

[23] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *International Conference on Machine Learning (ICML)*, pages 7197–7206, 2020.

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[26] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. BLOOM: A 176B-parameter open-access multilingual language model. *arXiv:2211.05100*, 2022.

[27] 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.

[28] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

[29] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv:2302.13971*, 2023.

[30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Towards accurate post-training quantization for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16026–16035, 2024.

[33] 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*, pages 38087–38099. PMLR, 2023.

[34] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[35] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv:2205.01068*, 2022.

Appendices

A Pseudo-code for the proposed *aespa*

In this appendix, we provide the pseudo-code for the proposed *aespa* excluded in the main text due to the page limitation.

Algorithm 1 Quantization

```
1: def QUANTIZATION($\boldsymbol{W}$,$\boldsymbol{X}$)
2:     Approximate the Hessian $\mathbf{H}$                                  ▷ See Table 4
3:     Estimate $\mathbb{E}[\boldsymbol{K}^T\boldsymbol{K}], \mathbb{E}[\boldsymbol{Q}^T\boldsymbol{Q}]$ for $\boldsymbol{W}_Q, \boldsymbol{W}_K$                   ▷ Table 4
4:     Set the step size $\boldsymbol{S}$ s.t. $\min_{\boldsymbol{S}} \operatorname{tr}\left(\Delta\boldsymbol{W}\mathbf{H}\Delta\boldsymbol{W}^T\right)$
5:     repeat
6:         Compute the Loss $\mathcal{L}$                                    ▷ Table 4
7:         Optimize $\boldsymbol{S}$ or $\boldsymbol{W}_{int}$ w.r.t $\mathcal{L}$ by certain algorithm
8:     until converged
9:     return $\boldsymbol{S}$ and $\boldsymbol{W}_{int}$                    ▷ step size and integer weight
```

Table 4: Row-wise Hessian $\mathbf{H}$ and quantization loss $\mathcal{L}$ for each layer

Layer	$\mathbf{H}$	$\mathcal{L}$
$\boldsymbol{W}_Q$	$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right]$	$\operatorname{tr}\left(\mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K}\right] \cdot \Delta\boldsymbol{W}\mathbf{H}\Delta\boldsymbol{W}^T\right)$
$\boldsymbol{W}_K$	$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right]$	$\operatorname{tr}\left(\mathbb{E}\left[\boldsymbol{Q}^T\boldsymbol{Q}\right] \cdot \Delta\boldsymbol{W}\mathbf{H}\Delta\boldsymbol{W}^T\right)$
$\boldsymbol{W}_V$	$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T\right]$	$\operatorname{tr}\left(\Delta\boldsymbol{W}\mathbf{H}\Delta\boldsymbol{W}^T\right)$
Others	$\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right]$	$\operatorname{tr}\left(\Delta\boldsymbol{W}\mathbf{H}\Delta\boldsymbol{W}^T\right)$

As mentioned, the proposed *aespa* consists of two main steps; *aespa* first determines the quantization parameters (*i.e.,* scale s and zero-point z in (3)) together with foldable parameters, as in [19, 13, 27, 20] (see line 4 in Algorithm 1), and then optimizes an integer weight $\boldsymbol{W}_{int}$ for each weight (see lines 5-8 in Algorithm 1). We emphasize that each iteration for the integer weight optimization can be performed efficiently based on pre-computed values (*i.e.,* $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$, $\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$, $\mathbb{E}[\boldsymbol{K}^T\boldsymbol{K}]$, and $\mathbb{E}[\boldsymbol{Q}^T\boldsymbol{Q}]$ in Table 4). We also note that while we have used Z-FOLD in computing the quantization parameters and used AdaRound in optimizing integer weights, our refined objectives in (17), (21), and (22) can be integrated into any layer-wise quantization scheme without effort because we only used the definition of the attention operation in our derivation.

B Validity of the proposed separate quantization strategy

Table 5: Performance (PPL ↓) of OPT-125M quantized with different strategies.

Method	Quantization Granularity	Reconstruction Target	W2A16		W3A16		W4A16	
			Wiki-2	C4	Wiki-2	C4	Wiki-2	C4
AdaRound	Layer-wise	Layer Output	160.7	95.63	35.44	31.86	29.51	27.78
BRECQ	Block-wise	Attention Output	60.38	47.85	33.25	29.74	28.86	27.43
Proposed	**Layer-wise**	**Attention Output**	69.23	51.92	32.89	29.75	28.98	27.42

We conduct experiments to demonstrate the importance of targeting attention-wise reconstruction and the validity of the proposed separate quantization strategy. In our experiments, we learn a weight-rounding policy using conventional AdaRound, but we set the loss function for each projection as the attention reconstruction error in (7) (not the layer reconstruction error; see Figure 2(c)).

Table 5 summarizes the quantization performance of AdaRound, BRECQ, and our approach on the OPT-125M model. As evident, our approach uniformly outperforms AdaRound for all bit-widths, although both methods quantize models layer-wise. This is because we can consider the cross-layer dependency (*i.e.,* the relationship between the query, key, and value) by targeting attention-wise reconstruction, which differs from AdaRound wherein layers are considered independent. Furthermore, once we target attention-wise reconstruction, separate layer-wise quantization does not incur severe performance degradation compared to the joint quantization method (BRECQ). Indeed, our approach causes only a marginal performance degradation for 2-bit and exhibits comparable performance for 3-bit and 4-bit.

One might wonder about the strategy of quantizing more than one layer jointly while maintaining remaining weights with full-precision, *e.g.,* simultaneous quantization of the query and key projections while fixing the value projection with full-precision. To say the conclusion first, in this case, we cannot use the proposed pre-computation-based loss computation strategy (see Section 3.4), resulting in a much longer quantization processing time. Specifically, when quantizing $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ simultaneously, the attention reconstruction error is expressed as

$$\Delta \mathrm{SA}_{Q,K} = \mathbb{E}\left[\left\|\mathrm{SA}(\widehat{\boldsymbol{Q}}, \widehat{\boldsymbol{K}}, \boldsymbol{V}) - \mathrm{SA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})\right\|_F^2\right] = \mathbb{E}\left[\|\Delta \boldsymbol{A}\boldsymbol{V}\|_F^2\right],$$

where

$$\Delta \boldsymbol{A} = \mathrm{softmax}\left(\frac{\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{K}}^T}{\sqrt{d}}\right) - \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right).$$

Then, by taking similar steps as in Section 3.3 (*i.e.,* approximating $\Delta \boldsymbol{A}$ with its first-order Taylor series and constructing an upper bound of $\Delta \mathrm{SA}_{Q,K}$), we can obtain the following objective:

$$\begin{aligned}
&\min_{\Delta \boldsymbol{W}_Q, \Delta \boldsymbol{W}_K} \mathbb{E}\left[\left\|\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{K}}^T - \boldsymbol{Q}\boldsymbol{K}^T\right\|_F^2\right] \\
&= \min_{\Delta \boldsymbol{W}_Q, \Delta \boldsymbol{W}_K} \mathbb{E}\left[\left\|\Delta \boldsymbol{Q}\boldsymbol{K}^T + \boldsymbol{Q}\Delta \boldsymbol{K}^T + \Delta \boldsymbol{Q}\Delta \boldsymbol{K}^T\right\|_F^2\right] \\
&= \min_{\Delta \boldsymbol{W}_Q, \Delta \boldsymbol{W}_K} \mathbb{E}\left[\left\|\boldsymbol{X}^T \Delta \boldsymbol{W}_Q^T \boldsymbol{K}^T + \boldsymbol{Q}\Delta \boldsymbol{W}_K \boldsymbol{X} + \boldsymbol{X}^T \Delta \boldsymbol{W}_Q^T \Delta \boldsymbol{W}_K \boldsymbol{X}\right\|_F^2\right].
\end{aligned}$$

Obviously, the objective becomes much more complex than the proposed ones in (15) and (16), and it would be difficult to simplify and accelerate the loss computation by exploiting pre-computed values as in *aespa*. In fact, without the proposed pre-computation-based loss computation, the simultaneous quantization of $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ requires 3.5 hours for the quantization of OPT-125M, which is about 44 times longer than the proposed *aespa* and even 1.9 times longer than BRECQ.

C Refined quantization objective (16) for the key projection

When quantizing the key projection $\boldsymbol{W}_K$, we fix the query and value projections with full-precision. In this case, the attention reconstruction error $\Delta\mathrm{SA}_K$ can be expressed as

$$\Delta\mathrm{SA}_K = \mathbb{E}\left[\left\|\mathrm{SA}(\boldsymbol{Q}, \widehat{\boldsymbol{K}}, \boldsymbol{V}) - \mathrm{SA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})\right\|_F^2\right] = \mathbb{E}\left[\|\Delta\boldsymbol{A}\boldsymbol{V}\|_F^2\right],$$

where

$$\Delta\boldsymbol{A} = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\widehat{\boldsymbol{K}}^T}{\sqrt{d}}\right) - \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right).$$

To avoid the computational overhead of repetitive softmax operation, we approximate $\Delta\boldsymbol{A}$ with its first-order Taylor series, which leads to

$$\Delta\mathrm{SA}_K \approx \frac{1}{d}\mathbb{E}\left[\left\|\boldsymbol{Q}\Delta\boldsymbol{K}^T\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\right\|_F^2\right] = \frac{1}{d}\mathbb{E}\left[\left\|\boldsymbol{Q}\Delta\boldsymbol{W}_K\boldsymbol{X}\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\right\|_F^2\right]. \tag{25}$$

Furthermore, to reduce the huge memory cost required to store the Jacobian $\mathbf{J}_{\mathrm{softmax}}$ having L^3 elements (see Footnote 5), we establish an upper bound of (25) and then use it as a surrogate of $\Delta\mathrm{SA}_K$. Specifically, we separate the term $\|\boldsymbol{Q}\Delta\boldsymbol{W}_K\boldsymbol{X}\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\|_F^2$ into two components as follows:

$$\left\|\boldsymbol{Q}\Delta\boldsymbol{W}_K\boldsymbol{X}\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\right\|_F^2 \leq \|\boldsymbol{Q}\Delta\boldsymbol{W}_K\boldsymbol{X}\|_F^2 \cdot \left\|\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\right\|_F^2.$$

Noting that the term $\left\|\mathbf{J}_{\mathrm{softmax}}^T\boldsymbol{V}\right\|_F^2$ is not affected by the quantization of $\boldsymbol{W}_K$ and thus fixed in the quantization process, we minimize $\|\boldsymbol{Q}\Delta\boldsymbol{W}_K\boldsymbol{X}\|_F^2$ to enforce $\Delta\mathrm{SA}_K$ to be small, which leads to the proposed objective in (16).

D Effectiveness of the proposed Hessian in (18)

We recall from Section 3.4 that the proposed quantization objective for the value projection is

$$\mathrm{tr}\left(\Delta \boldsymbol{W}_V \mathbb{E}\left[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T\right]\Delta \boldsymbol{W}_V^T\right),$$

which implies that the Hessian $\mathbf{H}_V$ for each row of $\boldsymbol{W}_V$ is

$$\mathbf{H}_V = 2\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T].$$

We note that the proposed Hessian $\mathbf{H}_V$ differs from

$$\mathbf{H} = 2\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T],$$

which has been commonly used as an approximated Hessian in existing methods [6, 7, 3, 13]. The key reason for the difference is that we consider the dependency between the query, key, and value projections by targeting attention-wise reconstruction, whereas the previous methods assumed independence.

To observe the effect of considering the cross-layer dependency, we use different Hessians (*i.e.,* $\mathbf{H}_V$ and $\mathbf{H}$) when quantizing language models via Z-FOLD and then compare the performance of the quantized models. As Table 6 shows, the quantization performance is much better when the proposed Hessian $\mathbf{H}_V$ is used, which demonstrates the importance of considering the cross-layer dependency.

Table 6: Quantization performance (PPL ↓) of Z-FOLD under different Hessians.

(a) WikiText-2

Hessian	Precision	OPT					LLaMA		
		125M	350M	1.3B	2.7B	6.7B	7B	13B	30B
$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ [6, 7, 3, 13]	INT3	39.59	25.97	16.10	13.54	11.65	6.756	5.708	4.931
$\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ **(ours)**		**35.05**	**24.81**	16.25	**13.40**	**11.43**	**6.529**	**5.669**	**4.693**
$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ [6, 7, 3, 13]	INT2	190.1	102.5	33.97	27.10	18.07	14.93	**13.03**	9.250
$\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ **(ours)**		**146.4**	**68.30**	**31.43**	**25.17**	**17.92**	**14.20**	13.15	**8.138**

(b) PTB

Hessian	Precision	OPT					LLaMA		
		125M	350M	1.3B	2.7B	6.7B	7B	13B	30B
$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ [6, 7, 3, 13]	INT3	53.08	39.23	22.73	20.18	16.64	11.73	**10.09**	8.979
$\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ **(ours)**		**49.88**	**37.62**	**22.66**	**19.78**	**16.55**	**11.39**	10.48	**8.657**
$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ [6, 7, 3, 13]	INT2	331.6	130.7	53.80	46.08	26.79	26.87	19.37	15.23
$\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ **(ours)**		**212.8**	**100.1**	**53.64**	**42.93**	**26.09**	**24.88**	**18.01**	**12.99**

(c) C4

Hessian	Precision	OPT					LLaMA		
		125M	350M	1.3B	2.7B	6.7B	7B	13B	30B
$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ [6, 7, 3, 13]	INT3	33.67	26.45	17.33	15.50	13.28	8.719	7.554	6.912
$\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ **(ours)**		**31.27**	**25.51**	**17.27**	**15.42**	**13.22**	**8.313**	**7.437**	**6.638**
$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ [6, 7, 3, 13]	INT2	125.3	71.37	31.67	25.99	19.79	16.88	14.61	11.90
$\mathbb{E}[\boldsymbol{X}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{X}^T]$ **(ours)**		**112.6**	**56.48**	**30.06**	**25.34**	**19.32**	**16.87**	**13.46**	**10.32**

E Proof of (19) and (21)

Note that $\mathbb{E}\left[\left\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2\right] = \mathbb{E}\left[\left\|\mathrm{vec}\left(\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right)\right\|_2^2\right]$, where $\mathrm{vec}(\cdot)$ denotes the vectorization operation. Then, by exploiting the following properties of Kronecker product

$$\begin{aligned}
\mathrm{vec}\left(\boldsymbol{ABC}\right) &= \left(\boldsymbol{C}^T \otimes \boldsymbol{A}\right)\mathrm{vec}(\boldsymbol{B}),\\
\left(\boldsymbol{A}\otimes\boldsymbol{B}\right)^T &= \boldsymbol{A}^T \otimes \boldsymbol{B}^T,\\
\left(\boldsymbol{A}\otimes\boldsymbol{B}\right)\left(\boldsymbol{C}\otimes\boldsymbol{D}\right) &= \boldsymbol{AC}\otimes\boldsymbol{BD},
\end{aligned}$$

we have

$$\begin{aligned}
\mathbb{E}\left[\left\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2\right] &= \mathbb{E}\left[\left\|\left(\boldsymbol{X}^T\otimes\boldsymbol{K}\right)\Delta\boldsymbol{w}_Q\right\|_2^2\right]\\
&= \mathbb{E}\left[\Delta\boldsymbol{w}_Q^T\left(\boldsymbol{X}^T\otimes\boldsymbol{K}\right)^T\left(\boldsymbol{X}^T\otimes\boldsymbol{K}\right)\Delta\boldsymbol{w}_Q\right]\\
&= \mathbb{E}\left[\Delta\boldsymbol{w}_Q^T\left(\boldsymbol{X}\otimes\boldsymbol{K}^T\right)\left(\boldsymbol{X}^T\otimes\boldsymbol{K}\right)\Delta\boldsymbol{w}_Q\right]\\
&= \mathbb{E}\left[\Delta\boldsymbol{w}_Q^T\left(\boldsymbol{X}\boldsymbol{X}^T\otimes\boldsymbol{K}^T\boldsymbol{K}\right)\Delta\boldsymbol{w}_Q\right] \qquad (26)\\
&= \Delta\boldsymbol{w}_Q^T\cdot\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\otimes\boldsymbol{K}^T\boldsymbol{K}\right]\cdot\Delta\boldsymbol{w}_Q,
\end{aligned}$$

which is the desired result in (19).

We now prove (21). By combining (19) and (20), we have

$$\mathbb{E}\left[\left\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2\right] \approx \Delta\boldsymbol{w}_Q^T\cdot\left(\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right]\otimes\mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K}\right]\right)\cdot\Delta\boldsymbol{w}_Q.$$

Note that since $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]$ and $\mathbb{E}[\boldsymbol{K}^T\boldsymbol{K}]$ are symmetric, there exist $\boldsymbol{G}_X$ and $\boldsymbol{G}_K$ such that

$$\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T] = \boldsymbol{G}_X\boldsymbol{G}_X^T,\ \mathbb{E}[\boldsymbol{K}^T\boldsymbol{K}] = \boldsymbol{G}_K^T\boldsymbol{G}_K.$$

Then, by following the steps used to derive (26) in the reverse order, we have

$$\begin{aligned}
\mathbb{E}\left[\left\|\boldsymbol{K}\Delta\boldsymbol{W}_Q\boldsymbol{X}\right\|_F^2\right] &= \Delta\boldsymbol{w}_Q^T\left(\boldsymbol{G}_X\boldsymbol{G}_X^T\otimes\boldsymbol{G}_K^T\boldsymbol{G}_K\right)\Delta\boldsymbol{w}_Q\\
&= \left\|\boldsymbol{G}_K\Delta\boldsymbol{W}_Q\boldsymbol{G}_X\right\|_F^2\\
&= \mathrm{tr}\left(\boldsymbol{G}_K\Delta\boldsymbol{W}_Q\boldsymbol{G}_X\boldsymbol{G}_X^T\Delta\boldsymbol{W}_Q^T\boldsymbol{G}_K^T\right)\\
&= \mathrm{tr}\left(\boldsymbol{G}_K^T\boldsymbol{G}_K\cdot\Delta\boldsymbol{W}_Q\cdot\boldsymbol{G}_X\boldsymbol{G}_X^T\cdot\Delta\boldsymbol{W}_Q^T\right)\\
&= \mathrm{tr}\left(\mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K}\right]\Delta\boldsymbol{W}_Q\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T\right]\Delta\boldsymbol{W}_Q^T\right),
\end{aligned}$$

which completes the proof of (21).

F Integration of proposed loss functions into existing PTQ schemes

We recall that we only utilized the definition of the attention operation when developing the proposed loss functions for the attention output reconstruction. Therefore, our loss functions can be integrated into any PTQ schemes based on layer-wise reconstruction and used to enhance their performance. In this section, we describe how to combine our loss functions with existing quantization schemes by taking AdaRound as an example.

In short, AdaRound learns a weight-rounding mechanism by solving the following optimization problem [23]:

$$\arg\min_{\boldsymbol{B}} \left\| \boldsymbol{W}\boldsymbol{X} - \widetilde{\boldsymbol{W}}\boldsymbol{X} \right\|_F^2 + \lambda \sum_{i,j} \left(1 - |2h(\boldsymbol{B}_{i,j}) - 1|^{\beta} \right), \tag{27}$$

where $\boldsymbol{B}$ is the continuous variable to be learned, h is the rectified sigmoid function, and $\widetilde{\boldsymbol{W}}$ is the soft-quantized weights defined as

$$\widetilde{\boldsymbol{W}} = s \cdot \text{clamp}\left(\left\lfloor \frac{\boldsymbol{W}}{s} \right\rfloor + h(\boldsymbol{B}), n, p \right).$$

One can see that the loss function of AdaRound consists of two components, layer-wise reconstruction error and weight-rounding loss.

To consider the cross-layer dependency between $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ in the learning process, we integrate the proposed loss functions developed for the attention output reconstruction into (27). In other words, we replace the layer-wise reconstruction error in (27) with our loss functions in (17), (21), and (22). For example, when learning the rounding policy for the query projection matrix $\boldsymbol{W}_Q$, the objective of the proposed *aespa* is expressed as

$$\arg\min_{\boldsymbol{B}_Q} \text{tr}\left(\mathbb{E}\left[\boldsymbol{K}^T\boldsymbol{K} \right] \Delta\boldsymbol{W}_Q \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^T \right] \Delta\boldsymbol{W}_Q^T \right) + \lambda \sum_{i,j} \left(1 - \left|2h(\boldsymbol{B}_{Q,i,j}) - 1\right|^{\beta} \right), \tag{28}$$

where $\Delta\boldsymbol{W}_Q = \boldsymbol{W}_Q - \widetilde{\boldsymbol{W}}_Q$.

G Complexity analysis for conventional block-wise quantization schemes

Recall from (7) that conventional block-wise quantization schemes require to compute $\mathrm{SA}(\widehat{\boldsymbol{Q}}, \widehat{\boldsymbol{K}}, \widehat{\boldsymbol{V}})$ in each iteration. This means that for each input sequence, one needs to perform

- forward pass for $\widehat{\boldsymbol{Q}}$, $\widehat{\boldsymbol{K}}$, and $\widehat{\boldsymbol{V}}$: $3d_h L(2d-1)$ flops
- matrix multiplications for computing $\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{K}}^T$ and $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{V}}$: $4d_h L^2 - d_h L - L^2$ flops
- softmax operation with additional scaling (*i.e.,* $\mathrm{softmax}(\frac{\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{K}}^T}{\sqrt{d_h}})$): $3L^2 + d_h L - L$ flops
- final computation of reconstruction error: $3d_h L - 1$ flops

If B input sequences are used in each quantization iteration, then the total number of flops required in conventional methods is

$$\mathcal{C}_{exist} = B(6d_h dL + 4d_h L^2 + 2L^2 - L - 1) = \mathcal{O}(Bd_h L \cdot \max\{d, L\}).$$

Comparison of $\mathcal{C}_{\textit{aespa}}$ and $\mathcal{C}_{exist}$ We now compare the complexities of *aespa* and conventional block-wise quantization methods in terms of the number of flops. Table 7 summarizes the computational costs required to quantize different sizes of OPT models. For conventional methods, we report the cost of using four sequences in each iteration ($B = 4$). We observe that the computational cost of *aespa* is considerably lower than that of conventional methods. In particular, for small-scale models (*e.g.,* OPT-125M, OPT-350M, and OPT-1.3B), *aespa* performs ten times fewer number of flops. One can notice that the gap between $\mathcal{C}_{aespa}$ and $\mathcal{C}_{exist}$ decreases as the model size increases. This is because the hidden size d exceeds the sequence length L (which is fixed for all models) as the model size increases. Nevertheless, *aespa* still incurs a lower computational cost, and the gap increases if conventional methods use larger batch sizes.

Table 7: Cost of *aespa* and conventional methods (GFLOPS)

	125M	350M	1.3B	2.7B	6.7B	13B
$\mathcal{C}_{exist}$	6.7	7.5	11	15	34	41
$\mathcal{C}_{aespa}$	0.24	0.42	1.6	3.2	13	20

H Comparison with block-wise PTQ schemes

We provide experimental results excluded from the main text due to page limitations.

Table 8: Performance (PPL ↓) of the proposed *aespa* and conventional block-wise PTQ methods.

(a) INT4 performance on WikiText-2 and C4

Dataset	Method	OPT				LLaMA			LLaMA2	
		125M	1.3B	2.7B	6.7B	7B	13B	30B	7B	13B
WikiText-2	FP16	27.65	14.63	12.47	10.86	5.677	5.091	4.101	5.472	4.884
	Brecq [18]	**28.86**	14.83	12.71	OOM	OOM	OOM	OOM	OOM	OOM
	OmniQuant [27]	30.42	15.15	12.89	11.20	5.907	5.256	4.263	5.850	5.064
	AffineQuant [20]	29.81	15.09	12.72	11.12	5.905	5.256	4.269	5.782	5.062
	aespa	28.87	**14.81**	**12.36**	**10.95**	**5.890**	**5.226**	**4.254**	**5.684**	**5.031**
C4	FP16	26.56	16.07	14.34	12.71	7.344	6.798	6.131	7.264	6.727
	Brecq [18]	27.43	16.42	14.61	OOM	OOM	OOM	OOM	OOM	OOM
	OmniQuant [27]	28.41	16.68	14.83	12.99	7.656	6.976	6.269	7.686	6.956
	AffineQuant [20]	28.04	16.58	14.74	12.92	7.654	6.974	6.270	7.644	6.927
	aespa	**27.24**	**16.31**	**14.55**	**12.82**	**7.633**	**6.945**	**6.256**	**7.508**	**6.891**

(b) Performance on PTB

Precision	Method	OPT				LLaMA		
		125M	1.3B	2.7B	6.7B	7B	13B	30B
FP16	Baseline	38.99	20.29	17.97	15.77	10.12	9.081	8.159
INT4	Brecq [18]	41.04	20.97	18.41	OOM	OOM	OOM	OOM
	OmniQuant [27]	42.34	21.32	18.70	16.04	10.57	9.330	8.354
	AffineQuant [20]	42.99	21.26	18.49	16.02	10.53	9.325	8.355
	aespa	**40.50**	**20.78**	**18.30**	**15.84**	**10.43**	**9.277**	**8.283**
INT3	Brecq [18]	46.93	23.41	19.82	OOM	OOM	OOM	OOM
	OmniQuant [27]	56.88	25.11	22.59	18.33	11.98	10.24	9.065
	AffineQuant [20]	51.47	24.38	21.03	17.40	11.92	10.24	8.998
	aespa	**44.96**	**22.35**	**19.48**	**16.28**	**11.45**	**9.818**	**8.684**
INT2	Brecq [18]	**90.22**	344.9	282.0	OOM	OOM	OOM	OOM
	OmniQuant [27]	NaN	377.9	2.0e3	7.7e3	33.51	NaN	17.38
	AffineQuant [20]	177.8	75.25	47.07	37.90	29.33	18.58	NaN
	aespa	99.12	**37.19**	**32.57**	**22.80**	**19.83**	**15.65**	**12.98**

* ‘NaN’ means that loss diverges in the quantization process.

* ‘OOM’ means that out-of-memory issues occur when quantizing models with a single A100 GPU.

I Comparison with layer-wise PTQ schemes

We provide experimental results excluded from the main text due to page limitations.

I.1 Results on OPT models

Table 9: Performance (PPL ↓) of *aespa* and existing layer-wise PTQ methods on OPT models.

(a) WikiText-2

Precision	Method	125M	350M	1.3B	2.7B	6.7B	13B	30B
FP16	Baseline	27.65	22.00	14.63	12.47	10.86	10.13	9.56
INT4	RTN	37.28	25.94	48.20	16.92	12.10	11.32	10.98
	OPTQ	32.49	23.68	15.50	12.85	11.12	10.33	9.670
	Z-Fold	31.03	23.08	15.00	12.47	11.01	10.21	9.537
	aespa	**28.87**	**22.55**	**14.81**	**12.36**	**10.95**	**10.18**	**9.511**
INT3	RTN	1.3e3	64.57	1.3e4	1.6e4	5.8e3	3.4e3	1.6e3
	OPTQ	52.95	33.29	20.36	16.94	13.01	11.65	10.44
	Z-Fold	39.59	25.97	16.10	13.54	11.65	10.62	9.902
	aespa	**32.71**	**24.45**	**15.79**	**13.14**	**11.23**	**10.52**	**9.760**
INT2	RTN	5.5e3	2.8e4	1.1e5	9.5e3	2.8e4	1.9e5	1.7e5
	OPTQ	4.1e3	1.1e4	8.3e3	9.3e3	2.0e3	539.8	56.63
	Z-Fold	190.1	102.5	33.97	27.10	18.07	33.48	13.48
	aespa	**71.18**	**54.89**	**24.26**	**22.22**	**15.71**	**15.27**	**11.91**

(b) PTB

Precision	Method	125M	350M	1.3B	2.7B	6.7B	13B	30B
FP16	Baseline	38.99	31.08	20.29	17.97	15.77	14.52	14.04
INT4	RTN	53.88	36.79	75.37	32.41	18.86	16.41	15.44
	OPTQ	46.54	33.27	21.74	19.04	16.42	14.88	14.21
	Z-Fold	44.17	33.51	20.96	18.45	15.98	14.65	14.11
	aespa	**40.50**	**32.17**	**20.78**	**18.30**	**15.84**	**14.65**	**14.09**
INT3	RTN	1.4e3	87.21	1.5e4	1.4e4	5.3e3	2.2e3	1.5e3
	OPTQ	74.07	46.10	29.76	25.06	19.22	16.42	15.08
	Z-Fold	53.08	39.23	22.73	20.18	16.64	15.23	14.60
	aespa	**44.96**	**36.15**	**22.35**	**19.48**	**16.28**	**15.06**	**14.43**
INT2	RTN	4.3e3	2.8e4	1.1e4	6.8e3	1.8e4	1.2e5	1.7e5
	OPTQ	3.5e3	1.2e4	6.6e3	8.0e3	2.5e3	458.4	83.81
	Z-Fold	331.6	130.7	53.80	46.08	26.79	79.69	20.39
	aespa	**99.12**	**79.86**	**37.19**	**32.57**	**22.80**	**23.93**	**17.51**

(c) C4

Precision	Method	125M	350M	1.3B	2.7B	6.7B	13B	30B
FP16	Baseline	26.56	22.59	16.07	14.34	12.71	12.06	11.44
INT4	RTN	33.88	26.21	27.50	18.83	14.37	13.32	13.55
	OPTQ	29.64	24.15	16.75	14.86	13.00	12.24	11.56
	Z-Fold	28.92	23.71	16.38	14.60	12.85	12.14	11.49
	aespa	**27.24**	**23.15**	**16.31**	**14.55**	**12.82**	**12.13**	**11.47**
INT3	RTN	834.4	55.15	6.6e3	1.2e4	5.0e3	2.8e3	1.8e3
	OPTQ	42.88	30.60	20.53	17.66	14.61	13.19	12.15
	Z-Fold	33.67	26.45	17.33	15.50	13.28	12.45	11.73
	aespa	**29.51**	**24.96**	**17.10**	**15.27**	**13.15**	**12.39**	**11.68**
INT2	RTN	3.7e3	1.6e4	7.7e3	7.7e3	1.4e4	9.7e4	5.8e4
	OPTQ	2.1e3	4.4e3	3.0e3	3.7e3	290.9	157.7	29.40
	Z-Fold	125.3	71.37	31.67	25.98	19.79	47.10	14.51
	aespa	**56.88**	**46.36**	**23.54**	**22.53**	**17.28**	**16.30**	**13.32**

I.2 Results on BLOOM models

Table 10: Performance (PPL ↓) of *aespa* and existing layer-wise PTQ methods on BLOOM models.

(a) INT4 performance on WikiText-2 and C4

Precision	Method	WikiText-2					C4				
		560M	1.1B	1.7B	3B	7.1B	560M	1.1B	1.7B	3B	7.1B
FP16	Baseline	22.42	17.69	15.39	13.48	11.37	26.60	22.05	19.49	17.49	15.20
INT4	RTN	25.82	19.98	16.96	14.75	12.09	29.80	24.42	21.24	18.75	16.05
	OPTQ	23.83	18.74	16.16	14.01	11.72	27.74	23.05	20.26	18.00	15.54
	Z-Fold	23.60	18.44	15.87	13.90	11.59	27.36	22.66	20.00	17.87	15.42
	aespa	**23.21**	**18.28**	**15.76**	**13.81**	**11.56**	**27.20**	**22.49**	**19.86**	**17.76**	**15.38**

(b) Performance on PTB

Precision	Method	560M	1.1B	1.7B	3B	7.1B
FP16	Baseline	43.69	57.96	30.00	25.34	20.83
INT4	RTN	50.96	66.79	33.52	27.65	22.40
	OPTQ	46.83	62.99	31.63	26.72	21.52
	Z-Fold	45.77	61.33	31.26	26.27	21.35
	aespa	**44.73**	**60.41**	**31.05**	**26.01**	**21.17**
INT3	RTN	124.8	184.0	105.5	66.24	34.94
	OPTQ	64.43	82.91	40.27	33.13	25.94
	Z-Fold	53.01	69.93	35.12	28.41	22.83
	aespa	**48.87**	**67.01**	**33.06**	**27.61**	**22.03**
INT2	RTN	7.4e5	1.1e6	2.5e5	1.2e5	2.2e5
	OPTQ	4.1e3	2.4e3	1.4e3	1.4e3	428.4
	Z-Fold	194.9	174.9	74.03	69.49	38.50
	aespa	**91.14**	**120.7**	**57.48**	**46.40**	**31.28**

I.3 Results on LLaMA models

Table 11: Performance (PPL ↓) of *aespa* and existing layer-wise PTQ methods on LLaMA models.

Precision	Method	WikiText-2			PTB			C4		
		7B	13B	30B	7B	13B	30B	7B	13B	30B
FP16	Baseline	5.677	5.091	4.101	10.12	9.081	8.159	7.344	6.798	6.131
INT4	RTN	6.291	5.525	4.536	11.25	9.775	8.653	8.121	7.232	6.537
	OPTQ	6.167	5.365	4.452	11.51	9.526	8.426	7.792	7.082	6.399
	Z-Fold	6.069	5.278	4.325	11.45	9.335	8.410	7.797	6.984	6.318
	aespa	**5.890**	**5.226**	**4.254**	**10.43**	**9.277**	**8.283**	**7.633**	**6.945**	**6.256**
INT3	RTN	25.61	11.78	14.87	98.89	28.94	28.79	30.86	14.46	30.04
	OPTQ	8.290	6.729	5.705	16.11	11.91	9.964	10.51	8.832	7.977
	Z-Fold	6.756	5.708	4.931	11.73	10.09	8.979	8.719	7.554	6.912
	aespa	**6.579**	**5.611**	**4.688**	**11.45**	**9.818**	**8.684**	**8.465**	**7.399**	**6.634**
INT2	RTN	1.1e5	5.7e4	2.7e4	9.9e4	8.1e4	3.3e4	1.1e5	5.9e4	2.8e4
	OPTQ	1.0e4	3.7e3	1.5e3	1.1e4	8.5e3	1.0e3	872.7	809.7	304.4
	Z-Fold	14.93	13.03	9.250	26.87	19.37	15.23	16.88	14.61	11.90
	aespa	**11.94**	**10.30**	**7.845**	**19.83**	**15.65**	**12.98**	**13.63**	**11.46**	**10.35**

I.4 Results on LLaMA2 models

Table 12: Performance (PPL ↓) of *aespa* and existing layer-wise PTQ methods on LLaMA2 models.

Precision	Method	WikiText-2		C4	
		7B	13B	7B	13B
FP16	Baseline	5.472	4.884	7.264	6.727
INT4	RTN	6.116	5.205	8.165	7.142
	OPTQ	6.060	5.158	7.838	7.030
	Z-Fold	5.815	5.099	7.602	6.996
	aespa	**5.684**	**5.031**	**7.508**	**6.891**
INT3	RTN	542.0	10.69	527.2	13.87
	OPTQ	8.664	6.554	11.24	8.761
	Z-Fold	6.606	5.710	8.666	7.692
	aespa	**6.241**	**5.462**	**8.225**	**7.392**
INT2	RTN	1.8e4	5.1e4	2.8e4	5.3e4
	OPTQ	7.5e3	2.1e3	1.7e3	560.7
	Z-Fold	20.79	15.56	21.98	16.90
	aespa	**13.99**	**12.14**	**14.36**	**13.59**

J Results for zero-shot tasks

We provide INT3 zero-shot performance results that are excluded from the main text due to page limitations.

Table 13: INT3 zero-shot performance (accuracy ↑) of *aespa* and existing methods.

Model	Method	ARC-c	ARC-e	HellaSwag	MMLU	Average
LLaMA-7B	FP16	44.62	72.85	76.18	32.19	56.46
	RTN	27.47	45.45	45.46	24.94	35.83
	OPTQ [7]	36.95	62.63	68.33	25.51	48.36
	Z-Fold [13]	41.21	66.92	72.50	28.90	52.38
	OmniQuant [27]	38.99	67.30	70.31	29.33	51.48
	AffineQuant [20]	39.25	65.61	70.56	29.68	51.28
	aespa	40.87	69.15	71.54	30.57	**53.03**
LLaMA-13B	FP16	47.87	74.75	79.08	43.46	61.29
	RTN	36.09	56.23	62.03	26.20	45.14
	OPTQ [7]	43.00	67.89	72.45	28.62	52.99
	Z-Fold [13]	44.88	71.00	75.66	36.88	57.11
	OmniQuant [27]	44.03	69.70	75.15	35.89	56.19
	AffineQuant [20]	43.60	70.24	75.10	32.67	55.40
	aespa	45.82	71.80	75.87	38.63	**58.03**
LLaMA-30B	FP16	52.90	78.96	82.63	54.66	67.29
	RTN	27.90	43.64	31.42	23.34	31.58
	OPTQ [7]	45.31	71.55	77.17	42.01	59.01
	Z-Fold [13]	50.34	75.84	79.69	51.00	64.22
	OmniQuant [27]	49.49	76.52	79.76	50.68	64.11
	AffineQuant [20]	49.66	77.10	79.49	50.37	64.16
	aespa	50.34	77.53	79.79	50.55	**64.55**
LLaMA2-7B	FP16	46.16	74.49	75.99	41.87	59.63
	RTN	25.94	35.48	35.39	23.14	29.99
	OPTQ [7]	37.46	63.01	64.85	28.79	48.53
	Z-Fold [13]	40.10	64.65	69.92	33.69	52.09
	OmniQuant [27]	40.36	67.30	71.00	31.26	52.48
	AffineQuant [20]	40.78	67.21	70.75	30.93	52.42
	aespa	41.38	69.11	71.78	38.18	**55.11**
LLaMA2-13B	FP16	49.06	77.44	79.39	52.10	64.50
	RTN	34.56	55.98	59.44	25.45	43.86
	OPTQ [7]	43.09	70.45	72.02	39.37	56.23
	Z-Fold [13]	46.42	72.77	74.79	47.91	60.47
	OmniQuant [27]	45.65	74.33	74.77	43.92	59.67
	AffineQuant [20]	47.18	75.42	75.28	45.61	60.87
	aespa	46.84	75.25	75.78	47.09	**61.24**

K Time and memory cost comparison

Table 14: Time and memory cost of *aespa* and existing methods

(a) INT2 quantization processing time

Target	Method	OPT				LLaMA		
		125M	1.3B	2.7B	6.7B	7B	13B	30B
layer-wise reconstruction	OPTQ [7]	0.66 min	0.08 hr	0.14 hr	0.29 hr	0.25 hr	0.45 hr	1.08 hr
	Z-FOLD [13]	1.09 min	0.27 hr	0.61 hr	2.58 hr	1.13 hr	2.48 hr	10.51 hr
attention-wise reconstruction	BRECQ [18]	108.2 min	10.71 hr	19.15 hr	OOM	OOM	OOM	OOM
	OmniQuant [27]	16.20 min	1.02 hr	1.63 hr	2.93 hr	2.37 hr	4.20 hr	9.84 hr
	AffineQuant [20]	28.33 min	2.57 hr	4.60 hr	9.85 hr	10.09 hr	18.76 hr	47.84 hr
	aespa	4.78 min	1.24 hr	2.83 hr	10.24 hr	6.84 hr	15.89 hr	53.69 hr

(b) Memory cost (GB)

Target	Method	OPT				LLaMA		
		125M	1.3B	2.7B	6.7B	7B	13B	30B
layer-wise reconstruction	OPTQ [7]	1.39	4.49	6.43	13.07	8.76	12.34	18.59
	Z-FOLD [13]	1.39	4.49	6.43	13.07	8.76	12.34	18.59
attention-wise reconstruction	BRECQ [18]	3.39	16.60	27.79	OOM	OOM	OOM	OOM
	OmniQuant [27]	1.94	5.87	7.09	11.68	12.61	17.02	24.53
	AffineQuant [20]	3.47	9.96	12.25	20.08	24.28	27.10	38.59
	aespa	1.68	5.47	6.84	12.26	21.69	29.27	43.00

* 'OOM' means that out-of-memory issues occur when quantizing models with a single NVIDIA A100 GPU.

Table 14 summarizes the processing time and memory cost of different quantization algorithms. We note that the processing time of the proposed *aespa* includes the time required for pre-computations (lines 2-4 in Algorithm 1).

As expected, *aespa* completes quantization much faster than BRECQ. For example, while BRECQ requires more than 10 hours to quantize OPT-1.3B, *aespa* completes quantization in 1.24 hours, which demonstrates the effectiveness of the proposed objectives and pre-computation-based loss computation strategy. Although other block-wise PTQ methods (OmniQuant/AffineQuant) perform quantization faster than *aespa* for hyper-scale models, they suffer from unstable training process or exhibit poor PPL performance (*e.g.,* PPL of OmniQuant is larger than 10^3 for OPT-6.7B; see Table 1). We also observe that OPTQ performs quantization very fast, but its PPL performance collapses completely regardless of the model size (see Table 9). Except *aespa*, Z-FOLD is the only method that shows both reasonable performance and processing time.

In real situations, when one needs to preserve the performance of the original model as much as possible, the proposed *aespa* would be an intriguing solution. In particular, when deploying LLMs on resource-constrained platforms where up to 7B models are commonly employed (*e.g.,* mobile devices), *aespa* would be a good fit. Even when fast quantization of hyper-scale models is needed, *aespa* can be used with a slight modification. Specifically, in time-limited cases, one can skip weight-rounding optimization (lines 5-8 in Algorithm 1) and simply perform the quantization parameter computation (line 4 in Algorithm 1) using the proposed Hessian that considers the cross-layer dependency (see (18)). In doing so, we can not only save the time required to perform weight-rounding learning, but also save the memory required to store pre-computed values ($\mathbb{E}[\mathbf{K}^T\mathbf{K}]$ and $\mathbb{E}[\mathbf{Q}^T\mathbf{Q}]$). Indeed, when performing only quantization parameter computation, we achieved a significant reduction in the processing time (see Table 15 below) while still exhibiting better performance than conventional methods (see Table 6 in Appendix D).

Table 15: INT2 quantization processing time of *aespa* without weight-rounding optimization

OPT				LLaMA		
125M	1.3B	2.7B	6.7B	7B	13B	30B
1.29 min	0.35 hr	0.74 hr	2.92 hr	1.47 hr	3.26 hr	12.50 hr

L Experimental results for different calibration datasets

One might wonder why the PPL performances of OmniQuant summarized in Table 1 are much worse than those reported in the original paper [27]; INT2 PPL performances of quantized LLaMA models are 18.18, NaN, and 10.15 for WikiText-2 in Table 1, which are worse than the values (15.47, 13.21, and 8.71) reported in [27]. This is because we used a different calibration dataset for quantization. Specifically, we used C4 when constructing a calibration dataset, while [27] used WikiText-2.

Additionally, we evaluate the performance of the proposed *aespa* using WikiText-2 as a calibration dataset. From Table 16, we observe that when calibration data are sampled from WikiText-2, our results for OmniQuant are comparable with those reported in the original paper [27]. While it has been reported that the performance variance of OmniQuant across different calibration datasets is low for INT3 and INT4 (see [27, Table A10]), such low variance does not hold for INT2. Furthermore, we observe that the proposed *aespa* outperforms OmniQuant regardless of the type of the calibration dataset.

Table 16: INT2 performances (PPL ↓) of *aespa* and OmniQuant for different calibration datasets

Calibration Dataset	Method	LLaMA		
		7B	13B	30B
C4	OmniQuant	18.18	NaN	10.15
	aespa	**11.94**	**10.30**	**7.845**
WikiText-2	OmniQuant	15.59	13.76	9.230
	aespa	**8.818**	**7.423**	**6.232**

* Test dataset: WikiText-2

M Quantization performance of *aespa* for high bit-widths

While previous results demonstrate that the proposed *aespa* is very competitive for low-bit quantization (*e.g.,* INT2 and INT3), one might wonder whether *aespa* can preserve the performance of the original full-precision model at high bit-widths. We thus evaluate INT4 and INT6 quantization performances of *aespa* with LLaMA models. From Table 17, we observe that *aespa* almost preserves the performance of the original full-precision model for the INT6 quantization. Even for the INT4 quantization, the performance degradation is very marginal (*e.g.,* less than 1% degradation for 13B and 30B models).

Table 17: INT4 and INT6 quantization performances of the proposed *aespa* (calibration data: C4)

Model	Precision	Perplexity (↓)		Zero-shot Accuracy (↑)				
		Wiki-2	C4	ARC-c	ARC-e	HellaSwag	MMLU	Average
LLaMA-7B	FP16	5.677	7.344	44.62	72.85	76.18	32.19	56.46
	INT4	5.896	7.602	43.77	71.51	74.90	31.33	55.38
	INT6	5.694	7.360	44.62	72.35	75.96	32.27	56.30
LLaMA-13B	FP16	5.091	6.798	47.87	74.75	79.08	43.46	61.29
	INT4	5.232	6.938	47.53	73.74	78.35	43.49	60.78
	INT6	5.096	6.809	48.04	74.96	78.98	43.24	61.31
LLaMA-30B	FP16	4.101	6.131	52.90	78.96	82.63	54.66	67.29
	INT4	4.260	6.254	52.99	78.16	82.28	53.62	66.76
	INT6	4.110	6.139	53.07	78.96	82.60	54.61	67.31

N Experimental results for different seeds

We recall that when constructing a calibration dataset, we randomly draw 128 sequences from the C4 dataset [24]. By changing the seed for the sampling, different calibration datasets can be constructed, which leads to different quantization results. In this appendix, we report the corresponding results and overall statistics.

Table 18: Quantization performance (PPL ↓) of *aespa* on OPT models for different seeds.

(a) WikiText-2

Precision	Seed	125M	350M	1.3B	2.7B	6.7B
INT4	0	28.87	22.55	14.81	12.36	10.95
	10	28.60	22.55	14.91	12.31	10.83
	100	28.75	22.85	14.94	12.35	10.90
INT3	0	32.71	24.45	15.79	13.14	11.23
	10	32.95	24.57	16.10	13.21	11.11
	100	33.38	24.45	15.70	13.27	11.24
INT2	0	71.18	54.89	24.26	22.22	15.71
	10	74.41	50.84	24.38	22.36	15.06
	100	77.03	53.12	25.93	22.39	15.66

(b) PTB

Precision	Seed	125M	350M	1.3B	2.7B	6.7B
INT4	0	40.50	32.17	20.78	18.30	15.84
	10	40.62	32.33	20.56	18.21	15.91
	100	40.11	32.60	20.55	18.20	15.86
INT3	0	44.96	36.15	22.35	19.48	16.28
	10	46.26	36.19	22.06	19.46	16.32
	100	47.54	35.61	22.10	19.66	16.39
INT2	0	99.12	79.86	37.19	32.57	22.80
	10	110.0	73.98	35.94	32.25	21.51
	100	106.0	79.09	37.33	31.90	21.86

(c) C4

Precision	Seed	125M	350M	1.3B	2.7B	6.7B
INT4	0	27.24	23.15	16.31	14.55	12.82
	10	27.23	23.13	16.32	14.54	12.81
	100	27.29	23.15	16.34	14.54	12.81
INT3	0	29.51	24.96	17.10	15.27	13.15
	10	29.59	24.98	17.06	15.29	13.15
	100	29.58	25.00	17.09	15.37	13.15
INT2	0	56.88	46.36	23.54	22.53	17.28
	10	56.23	44.02	23.91	22.56	16.91
	100	56.78	45.21	24.41	22.42	17.30

Table 19: Quantization performance statistics (PPL ↓) of *aespa* on OPT models.

Precision	Dataset	125M	350M	1.3B	2.7B	6.7B
INT4	Wiki-2	28.74 ± 0.139	22.65 ± 0.172	14.89 ± 0.066	12.34 ± 0.023	10.89 ± 0.058
	PTB	40.41 ± 0.264	32.36 ± 0.217	20.63 ± 0.128	18.24 ± 0.057	15.87 ± 0.034
	C4	27.25 ± 0.036	23.14 ± 0.014	16.33 ± 0.016	14.55 ± 0.005	12.81 ± 0.002
INT3	Wiki-2	33.01 ± 0.340	24.49 ± 0.068	15.87 ± 0.209	13.21 ± 0.064	11.19 ± 0.068
	PTB	46.26 ± 1.287	35.98 ± 0.321	22.17 ± 0.159	19.54 ± 0.109	16.33 ± 0.058
	C4	29.56 ± 0.043	24.98 ± 0.024	17.08 ± 0.021	15.31 ± 0.050	13.15 ± 0.004
INT2	Wiki-2	74.20 ± 2.931	52.95 ± 2.029	24.86 ± 0.930	22.32 ± 0.088	15.48 ± 0.363
	PTB	105.0 ± 5.495	77.64 ± 3.195	36.82 ± 0.766	32.24 ± 0.335	22.06 ± 0.667
	C4	56.63 ± 0.350	45.20 ± 1.171	23.95 ± 0.438	22.50 ± 0.076	17.17 ± 0.219

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our main contribution (proposal of a next-level quantization algorithms that balance accuracy and efficiency for the quantization of hyper-scale Transformer models).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations together with some future research directions in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We have included detailed derivations and complete proofs of the proposed quantization loss functions in Sections 3.3, 3.4, 3.5 and Appendices C, E, F, G.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have included detailed instructions, e.g., pseudo-code for the proposed algorithm (see Algorithm 1), hyperparameter settings, and stopping criterion (see Section 4), to reproduce the main experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code will be available at `https://github.com/SamsungLabs/aespa`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified all the training details including hyperparameter settings and stopping criterion in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have reported the results accompanied by statistics (e.g., mean and standard deviation) in Appendix N.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have mentioned that we used a single NVIDIA A100 GPU (80 GB) in our experiments (see the last sentence in Section 4.1).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research have been conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed potential positive societal impacts of our work in Section 1, Section 4, and Section 5. As evident from our experiments, the proposed algorithm can serve as a useful quantization solution that pursues both efficiency and accuracy when deploying LLMs on resource-constrained devices.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no risks for misuse (e.g., pretrained language models, image generators, or scraped datasets).

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the original paper and the codes provided by the authors of the original paper as in Footnote 8.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.