Outlier Suppression: Pushing the Limit of Low-bit Transformer Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Transformer architecture has become the fundamental ingredient of the widespread natural language processing (NLP) models. With the trends of large NLP models, the increasing memory and computation costs hinder their efficient deployment on resource-limited devices. Therefore, transformer quantization attracts wide research interest. Recent works recognize that the outliers in some special tokens are the critical bottleneck for the quantization accuracy. However, their solution does not tackle it from the origin but walks around it with an increased computation cost. To fundamentally address this problem, this paper delves into the inherent inducement and importance of the outliers and discovers that γ in LayerNorm (LN) acts as a sinful amplifier for the outliers, and some outliers from a few tokens can be sharply clipped without negative impacts. Motivated by these findings, we propose an outlier suppression framework to overcome the quantization bottleneck of Transformer language models, including Gamma Migration and then Token-Wise Clipping. Gamma Migration utilizes migration equivalence to move the outlier amplifier to subsequent branches without any extra computation cost, avoiding the amplification of outliers and contributing to a more quantization-friendly distribution. Token-Wise Clipping takes the large variance of token range into consideration and clips the unimportant values with high efficiency in a token-wise coarse-to-fine pipeline. This framework effectively suppresses the outliers and can be used in a plug-and-play mode. Extensive experiments prove that our outlier suppression methods surpass the existing works and, for the first time, push the 6-bit post-training BERT quantization to the full-precision (FP) level.

1 Introduction

Transformer [1] has been one of the most common architectures in natural language processing along with lots of popular self-supervised models, such as BERT [2], RoBERTa [3], XLNet [4] and BART [5]. While these pre-trained models have demonstrated a significant superiority in performance, the memory and computation overheads have been a well-known concern, peculiarly in the real development. Therefore, model compression has attracted much attention from both academia and industry. Among them, quantization, working in the low-precision arithmetic fashion, is one of the key approaches for compressing large models and fitting them into the lightweight devices.

These days, more interest has been attracted to quantization of Transformer-based models. [6] proposes an 8-bit quantization scheme for BERT-like models. [7] advises a group-wise quantization technique and analyzes the limit of mixed-precision using second-order Hessian information. [8, 9] combine distillation [10] with quantization. [11] explores the availability of integer-only quantization with the approximation of nonlinear operations. And [12] utilizes randomness to reduce the induced

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

bias of Straight Through Estimation during training. Nonetheless, few studies investigate the inherent bottleneck of quantizing Transformer-based models.

Recently, some papers [13, 14] indicate that there exist significantly larger outliers in NLP models than in the computer vision ones. And these extreme outliers (some close to 100) behave in structured patterns, bringing devastating damage to the quantization accuracy (e.g., a 12% drop even for the 8-bit). For this critical outlier problem, existing method [13] chooses bypassing solutions such as a finer quantization granularity. However, this scheme causes an increased computation cost and unavoidably hinders the acceleration effect.

In this paper, to suppress the outliers rather than walk around them, we make an in-depth analysis to investigate the inducement of the outliers and the impact of clipping the outliers. Specifically, we first exploit the inducement and find that the scaling parameter $\boldsymbol{\gamma}$ in the LayerNorm structure works as an outlier amplifier and strengthens the outliers in the output. By extracting it, the activation is more robust to quantization. Then we further study the impact of outlier clipping and discover that different outliers may have different impacts on the full-precision performance when they are clipped. More interestingly, the more aggressive outliers provided by a few tokens, such as the separator token, can be clipped sharply and safely without much accuracy degradation.

Motivated by these findings, we propose an outlier suppression framework to push the limit of low-bit Transformer language models, which suppresses the outliers by equivalently migrating the outlier amplifier and efficiently finding an appropriate clipping range. Such framework contains two key components: Gamma Migration and Token-Wise Clipping, corresponding to these two findings. Gamma Migration extracts the scaling parameter $\boldsymbol{\gamma}$ in LayerNorm and transfers it into subsequent modules with an equivalent transformation, significantly alleviating the outliers. Thus we can quantize on a more robust activation with no extra computation overhead. Then, Token-Wise Clipping further suppresses the unimportant outliers which still exist without the amplification. Compared to previous calibration methods, which ignore the outliers' importance or suffer from large time costs on those useless outliers, our method leverages the fact that those less important values only belong to a few tokens and first obtains a preliminary clipping range from a token perspective, then optimizes it in a fine-grained way. Our proposed framework can be combined with existing methods, and the thought of outlier suppression shall shed new light on the study of NLP quantization.

To summarize, our contributions are as follows:

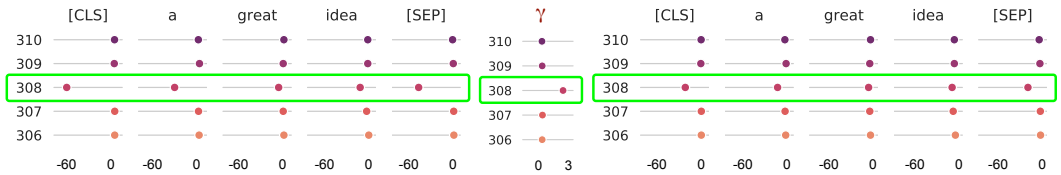
1. We delve into the inducement and clipping impact of outliers in the NLP models and draw two critical findings that are helpful for handling the bottleneck of Transformer Quantization.
2. Based on the findings, an outlier suppression framework containing Gamma Migration and Token-Wise Clipping is proposed. This framework is efficient, easy to implement and plug-and-play.
3. Gamma Migration suppresses the outliers from the inducement aspect. It transfers the outlier amplifier in LayerNorm to the subsequent modules utilizing an equivalence transformation, contributing to a quantization-friendly distribution without any extra inference time.
4. Token-Wise Clipping scheme suppresses the outliers from the importance aspect. It skips over those unimportant outliers quickly from the token perspective and focuses on the influential area with fine-grained learning.
5. Extensive experiments on various NLP models (BERT, RoBERTa, BART) and tasks (text classification, question answering and summarization) prove that our outlier suppression framework sets up a new state of the art for transformer quantization, and for the first time, pushes the 6-bit post-training quantization (PTQ) and 4-bit quantization-aware training (QAT) accuracy of BERT to the full-precision level.

2 Preliminaries

Basic Notations. We mark matrices as $\boldsymbol{X}$, and vectors as $\boldsymbol{x}$. Operator $\cdot$ denotes the scalar multiplication, and $\odot$ is adopted for element-wise multiplication on matrices or vectors. Also, we use $\boldsymbol{W}\boldsymbol{x}$ as matrix-vector multiplication. Specifically, in NLP tasks refer to tokens, $\boldsymbol{X}_{t,j}$ stands for the element at token t and embedding j, and $\boldsymbol{x}_t$ represents the embedding of token t.

Quantizer. Quantization usually includes two operations.

$$\bar{x} = clip(\lfloor \frac{x}{s} \rceil + z,\, 0,\, 2^b - 1), \quad \hat{x} = (\bar{x} - z) \cdot s \tag{1}$$



(a) $\widetilde{\boldsymbol{X}}$ (b) $\boldsymbol{\gamma}$ (c) $\boldsymbol{X}'$

Figure 1: Presentation of outliers over $\widetilde{\boldsymbol{X}}$, $\boldsymbol{\gamma}$ and $\boldsymbol{X}'$ of LayerNorm on BERT-SST-2. For example, at dimension 308, $\boldsymbol{\gamma}$ and $\widetilde{\boldsymbol{X}}$ both have sharper values. By excluding $\boldsymbol{\gamma}$, it can be seen that $\boldsymbol{X}'$ holds milder distribution than $\widetilde{\boldsymbol{X}}$. More evidence is put in Sec. D.1.

where s (step size), z (zero point) are quantization parameters, b is the bit setting. The first operation called "Quant" maps continuous numbers (x) to discrete points ($\bar{x}$) for integer-arithmetic-only matrix computation. The second operation called "DeQuant" recovers it to $\hat{x}$ after multiplication.

3 Outlier analysis

For Transformer-based models, standard 6/8-bit PTQ or 4-bit QAT would cause severe accuracy degradation. Investigating each quantizer, we recognize that the output of LayerNorm structures and GELU functions hold some sharp outliers, which lead to the large quantization error. Evidence and experimental results in Sec. B.2.

These outliers show some structured characteristics that they emerge at some embedding dimensions (Fig. 9). Upon these embeddings, some tokens even hold more aggressive outliers (Fig. 10). To suppress these harmful outliers for quantization, we explore the underlying inducement, and also the clipping impact of them.

3.1 Inducement of outliers

During the exploration, we find that the scaling parameter in LayerNorm amplifies the outliers in the embedding dimensions. And the phenomenon that some tokens even have sharper outliers might correlate with token frequency during pre-training (See Sec. C.2). In this part, we mainly explain the first inducement for solving these outliers from the origin. For another one, account for improving the quantization accuracy with as low costs as possible, it is unrealistic to adjust the pre-training and we discuss the clipping impact in next part to solve these outliers from clipping perspective.

Concerning about the challenges in quantizing the LayerNorm, a natural action is to dive into its internal structure. For token t at j^{th} embedding dimension, it first normalizes the input using mean ($\boldsymbol{u}_t$) and variance ($\boldsymbol{\sigma}_t^2$) each forward pass, then scales and shifts the value with parameter $\boldsymbol{\gamma}_j$ and $\boldsymbol{\beta}_j$.

$$\textbf{LayerNorm}: \quad \widetilde{\boldsymbol{X}}_{t,j} = \frac{\boldsymbol{X}_{t,j} - \boldsymbol{u}_t}{\sqrt{\boldsymbol{\sigma}_t^2 + \epsilon}} \cdot \boldsymbol{\gamma}_j + \boldsymbol{\beta}_j \tag{2}$$

Thus, we observe the parameter distribution of LayerNorm and surprisingly find that at the same outlier dimensions with the output, multiplier $\boldsymbol{\gamma}_j$ holds sharper values than others (Fig. 1b). Besides, the range of the adder $\boldsymbol{\beta}_j$ is much smaller (e.g., (0, 3)) than its left part (e.g., (-25, 0)), so we ignore it for determining the key point. That is to say, $\boldsymbol{\gamma}$ might be a core ingredient for the situation Fig. 1a, especially can contribute to outliers across tokens as a shared parameter.

This phenomenon enlightens us to remove the influence of $\boldsymbol{\gamma}$ by extracting it from Eq. (2) and see the distribution of tensor $\boldsymbol{X}'$.

$$\textbf{Non-scaling LayerNorm}: \quad \boldsymbol{X}'_{t,j} = \frac{\boldsymbol{X}_{t,j} - \boldsymbol{u}_t}{\sqrt{\boldsymbol{\sigma}_t^2 + \epsilon}} + \frac{\boldsymbol{\beta}_j}{\boldsymbol{\gamma}_j} \tag{3}$$

Comparing Fig. 1a and Fig. 1c, it is obviously that $\boldsymbol{X}'$ denotes milder distribution with weaker outliers and reveals that parameter $\boldsymbol{\gamma}$ does strengthen the outliers.

To quantitatively illustrate that tensor $\boldsymbol{X}'$ behaves more robust than $\widetilde{\boldsymbol{X}}$ in quantization, we adopt the cosine similarity metric to evaluate the quantization information loss. From Table 1, the second row with higher similarity and thus less quantization error explains that the quantization performance can be improved using Non-scaling LayerNorm.

Tensor	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
$\widetilde{\boldsymbol{X}}$	97.16	97.03	97.61	94.37	93.41	93.53	93.31	93.61	94.56	95.62	96.13	98.57
$\boldsymbol{X}'$	**99.23**	**99.22**	**99.11**	**99.02**	**98.99**	**99.00**	**98.99**	**98.83**	**98.70**	**99.05**	**99.44**	**99.07**

Table 1: Cosine similarity (%) of the quantized value (6-bit) and the real signal for $\widetilde{\boldsymbol{X}}$ and $\boldsymbol{X}'$ across 12 Attn-LN (LayerNorm after Multi-Head Attention) layers on BERT-SST2. Higher is better. More evidence in Sec. D.1.

3.2 Impact of outlier clipping

In this part, we investigate the impact of clipping the outliers. Our conclusion is that the more aggressive outliers can be clipped without affecting the performance on FP models, and these little impact outliers correspond to only several tokens. This indicates us to pursue an appropriate clipping range with consideration on the outliers' importance for better trade-off between the clipping error and the rounding one.

Accuracy impact of outlier clipping. We take the outliers after GELU as an example here, while a similar phenomenon can also be found in LayerNorm's output (Sec. D.2). Fig. 2 shows that the more striking outliers with signal even at 100 can be clipped sharply even to 10, with accuracy still staying at 91.02 in the FP model, while accuracy drops rapidly to 85.93 with too many outliers cut.

Token impact of outlier clipping. Motivated by [13], they refer that the separator token [SEP] attends to larger values. We are also aware of the different ranges provided by different tokens. By drawing the red points in Fig. 2, which calculates the proportion of clipped tokens, it can be clearly seen that the more aggressive area covers a lot from 10 to 100 but only matches with only 3% tokens. Destroying those sharper outliers belonging to several tokens will not affect the performance.

Since the outliers in the NLP models show markedly different importance and the large variance of token ranges causes severely long tail distribution, the methods ignoring importance [15] and requiring hyper-parameter tuning [16] fail to find a suitable clipping range.

Fortunately, combined with the observation that the long tail part is only filled with a few tokens, we introduce a method in Sec. 4.2 to leverage the token's indication to quickly skip over those unimportant areas and reach a favourable quantization clipping value.

4 Method

In this section, we introduce our techniques based on the above analysis to break the outlier bottleneck. To solve the outlier amplification problem, we devise the Gamma Migration, which transfers the

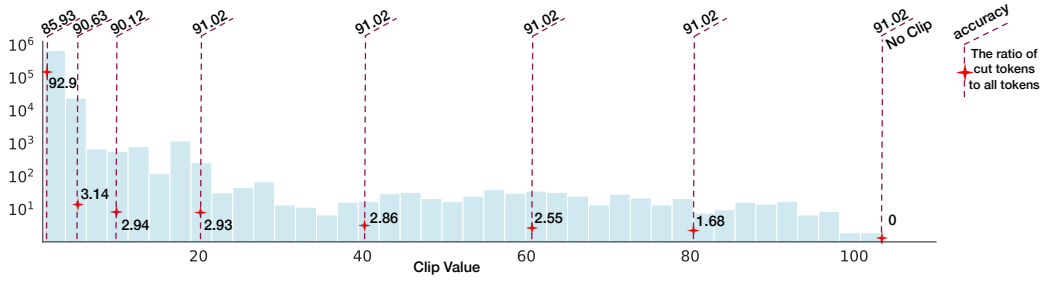


Figure 2: To detect the impact of clipping the outliers, we first draw the distribution using (mean + 3 * std) as its left border, then enumerate the value to cut the tensor on RoBERTa-QNLI. Red points reflect the proportion of clipped tokens. More evidence in Sec. D.2.

gamma in LayerNorm into subsequent modules. Then, the Token-Wise Clipping further suppresses the unimportant outliers which still exist without the amplification effectively and efficiently.

4.1 Gamma Migration

As pointed out in Sec. 3.1, activation without going through the scaling parameter produces less quantization error. In this way, we split the LayerNorm function, migrate $\boldsymbol{\gamma}$ into follow-up structures and quantize the output of the Non-scaling LayerNorm. The transformation is equivalent for the FP model and brings more robust activation for the low-bit one. The overall flow is illustrated in Fig. 3.

Migration equivalence on FP model. Naturally, as refered in Eq. (3), we extract the parameter $\boldsymbol{\gamma}$ and transform the LayerNorm into Non-scaling one, thus seperate $\boldsymbol{X}'_{t,j}$ from $\widetilde{\boldsymbol{X}}_{t,j}$.

$$\widetilde{\boldsymbol{X}}_{t,j} = \boldsymbol{X}'_{t,j} \cdot \boldsymbol{\gamma}_j \tag{4}$$

Since the residual connection is frequently adopted after LayerNorm ([17, 18, 19]), it is necessary to consider moving the parameter $\boldsymbol{\gamma}$ into two branches. To put it in practical terms, such as LayerNorm after Multi-Head Attention (Fig. 3), we cancel out the parameter in LayerNorm, establish $\boldsymbol{\gamma}$ on the shortcut branch and inject the value into the weight of next Layer.

Next, we show how the weight absorbs $\boldsymbol{\gamma}$. For linear layers, we have the following equation:

$$\boldsymbol{W}(\boldsymbol{x} \odot \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \cdots \\ \gamma_n \end{bmatrix}) = (\boldsymbol{W} \odot \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_n \\ \gamma_1 & \gamma_2 & \cdots & \gamma_n \\ \cdots & & & \\ \gamma_1 & \gamma_2 & \cdots & \gamma_n \end{bmatrix})\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{x}$ serves as a column vector and $\boldsymbol{\gamma} \in \mathbb{R}^n$. The proof is available in Appendix A. Because $\boldsymbol{\gamma}$ is a shared parameter, Eq. (5) holds for each token's embedding, and we can transfer the $\boldsymbol{\gamma}$ into the next layer's weight, successfully.

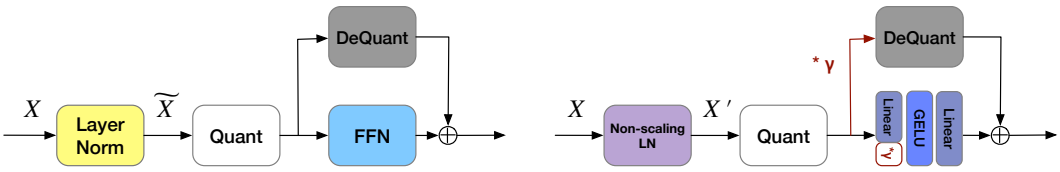


Figure 3: Comparison of the quantization flow before (left) and after (right) Gamma Migration. The original LayerNorm = the Non-scaling LayerNorm * $\boldsymbol{\gamma}$. The migrated $\boldsymbol{\gamma}$ can be fused into the subsequent DeQuant and Linear layers without any extra computation cost. Other detailed applications such as LayerNorm in encoder-decoder structure, see Fig. 6, Fig. 7.

Quantization after migration. Based on above equivalent transformation, we clarify the quantization pattern of the Non-scaling LayerNorm. As shown in Fig. 3, the "Quant" process is employed at $\boldsymbol{X}'$, then the output in one branch enjoys the matrix multiplication with the quantized altered weight, in another branch multiplies parameter $\boldsymbol{\gamma}$ and experiences the "DeQuant" process. In fact, this means delaying the $\boldsymbol{\gamma}$ calculation. Hence, this new design will not increase the computation overhead.

Effect of migration. We then analyze the effect of $\boldsymbol{\gamma}$ migration brought to weight and activation to illustrate that the activation quantization burden has been greatly alleviated with relatively a slight influence on weight. As presented in Fig. 1, outliers emerge at the same embeddings on $\boldsymbol{\gamma}$, activation before ($\boldsymbol{X}'$) and after ($\widetilde{\boldsymbol{X}}$) scaling function. In the original structure, the absolute max range of output can be actually rewrite as $|max(\boldsymbol{X}')| * |max(\boldsymbol{\gamma})|$. However, the weight matrix does not have the same embedding outlier phenomenon as the activation. Thus, in our method, the newly quantized activation range becomes $|max(\boldsymbol{X}')|$ and weight range will not be amplified $|max(\boldsymbol{\gamma})|$ times. Experimentally, Table 1 in Sec. 3.1 has validated the favor on activation. We also calculate the cosine similarity for the changed weight and observe that $\boldsymbol{\gamma}$ has little impact on weight (Table 2). By the way, quantization-aware training is also able to enjoy the benefit of Gamma Migration (Fig. 5).

4.2 Token-Wise Clipping

As illustrated empirically in the analysis, the more serious outliers characterized by several tokens can be clipped safely while others can not. To find a proper clipping range under limited bit-width,

Tensor	0	1	2	3	4	5	6	7	8	9	10	11
original weight	99.95	99.95	99.95	99.95	99.95	99.95	99.95	99.95	99.95	99.95	99.95	99.95
changed weight	99.95	99.95	99.95	99.90	99.90	99.92	99.94	99.95	99.95	99.95	99.91	99.94

Table 2: Cosine similarity (%) between the quantized value (6-bit) and the real signal for original weight and the changed weight across 12 Intermediate layers on BERT-SST2. It can be seen that there is little disparity between the two rows, especially compared with Table 1.

we propose a Token-Wise Clipping method to show how to utilize the token's information to jump over the relatively unimportant outliers and focus on the key ones.

Regarding the very different clipping impact of outliers, one understanding is to take the final loss into consideration. Thus, we first give the quantization loss definition below and minimize it using a coarse-to-fine paradigm.

$$L(s) = \|\hat{f}(s) - f\|_F^2, \tag{6}$$

where s is the step size parameter in the quantized model, and the loss represents the distance between the final quantized output $\hat{f}(s)$ and the real one f.

Coarse-grained Stage. At this stage, our aim is to quickly skip over the area with little impact after clipping and access to the critical area. Sec. 3.2 explains that the long tail only matches with a few tokens. Therefore, we suggest using the max value of the embedding at token t to be its representatives (min value as representatives for negative outliers). A new tensor with T elements can be constructed by taking out the maximum signal for each token:

$$\boldsymbol{o}^u = \{max(\boldsymbol{x}_1),\ max(\boldsymbol{x}_2),\ \dots,\ max(\boldsymbol{x}_T)\}, \tag{7}$$

where $\boldsymbol{o}^u$ is marked as the collection of upper bounds, $\boldsymbol{o}^l$ as collection of lower bounds. Then for a clipping ratio α on $\boldsymbol{o}^u$, calculate the corresponding clipping value c^u, and use it to cut the tensor.

$$c^u = quantile(\boldsymbol{o}^u, \alpha). \tag{8}$$

where the quantile function computes the α-th quantiles of the sorted $\boldsymbol{o}^u$.

Through grid search of token-wise clipping ratio, we get step size $s = \frac{c^u - c^l}{2^b - 1}$ (b is the bit-width), and take the one with minimal quantization loss Eq. (6). The initialized step size is marked as s_0 for the fine-grained stage.

Fine-grained Stage. With s_0 in the coarse phase, the learning procedure is equipped to make some fine-grained adjustments. In detail, we optimize parameter s towards Eq. (6) using gradient descent with initialization s_0 and small learning rate η.

$$s = s - \eta \frac{\partial L(s)}{\partial s} \tag{9}$$

Benifits. Along with travelling over the representative of tokens in the first step, the long tail can be passed quickly Fig. 4. In the second step, at a good initialization point of loss surface, learning adjustments further provide a guarantee for the final effect. Moreover, by virtue of the reduced tensor ($\boldsymbol{o}^u$ distilled from $\boldsymbol{X}$), it runs very fast each iteration. And the whole grid search is more efficient than OMSE [15] here. Ablation study in Sec. 5.2 demonstrates the enjoyable performance at the first step, and Sec. D.3 gives comparisons among our scheme and other existing approaches.

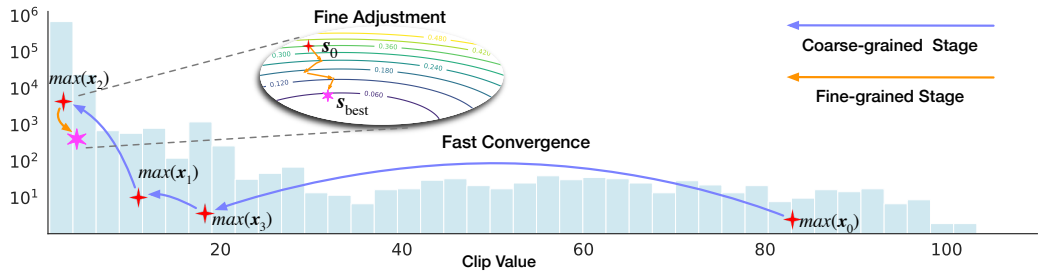


Figure 4: Flow diagram of the proposed Token-Wise Clipping

5 Experiments

In this section, we evaluate our proposed framework from two aspects. Sec. 5.2 shows the effect of Gamma Migration and Token-Wise Clipping, respectively. In Sec. 5.3, we evaluate the overall method across classification, question answering, and summarization tasks on BERT, RoBERTa and BART models. Here, 4-4-4 presents 4-bit weights, embeddings and activations.

5.1 Setup

Datasets. We conduct experiments on GLUE [20] tasks on both PTQ and QAT settings. Besides, more study is done on SQuAD [21, 22], XSum [23] and CNN/DailyMail [24] to further validate the robustness of our approach.

Baseline. For PTQ algorithm, we implement the prevalent techniques including MinMax [25], OMSE [15], Percentile [26], EasyQuant [27] and PEG [13]. For QAT algorithm, we present the results of Q-BERT [7], Q8BERT [6], PEG [13]. Because our framework applies at the calibration phase of QAT and we combine it with strong training baseline LSQ+[28], we also show the results of the pure LSQ+, and another canonical quantization approach PACT [29]. we also evaluate how our method performs combined with the knowledge distillation (KD) in TernaryBERT [8].

Implementation details. We give a detailed description of the quantization scheme and the quantization nodes in Sec. B.1, which is more friendly to hardware than some existing papers ([9, 8]) adopt. For PTQ experiments, we sample 256 examples as the calibration dataset and set batch size as 32. For QAT experiments on GLUE benchmark, we equip our methods with LSQ+ [28], providing a good calibration initiliazation. About hyper-parameters, learning rates are searched for both the baseline mechanisms and our methods. Details in Appendix F.

5.2 Ablation Study

In this subsection, we ablate the design elements in the proposed framework (Table 3). As a general plug-in module, Gamma Migration helps both the MinMax and Token-Wise Clipping. And the Token-Wise Clipping also surpasses the baseline by a large margin: 17.53% on QNLI, 12.5% on MRPC (comparison with other calibration algorithms see Sec. D.3). About the phenomenon that the fine-grained stage sometimes doesn't improve much upon the coarse-grained one, we think it's due to the already good enough result produced by the coarse step.

Besides PTQ, Fig. 5 shows that with our method, the training of QAT becomes faster and easier.

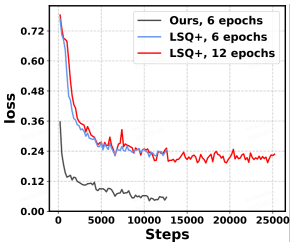


Figure 5: QAT fine-tuning process on BERT-SST-2.

Method	CoLA (Matt.)	MNLI (acc m/mm)	MRPC (f1/acc)	QNLI (acc)	QQP (f1/acc)	RTE (acc)	SST-2 (acc)	STS-B (Pear./Spear.)
FP32	62.50	87.75/87.23	93.1/90.44	92.68	88.78/91.6	80.51	95.18	91.04/90.72
Baseline (MinMax)	0.0	34.9/35.0	71.64/67.4	62.13	51.88/74.37	49.82	77.87	44.11/46.74
MinMax + Gamma Migration	0.0	53.53/54.64	**87.97/82.84**	78.56	78.04/85.3	55.6	85.67	61.03/63.22
Token-Wise Clipping (Coarse)	34.95	80.56/80.84	85.05/79.41	79.46	85.96/89.31	66.43	91.63	82.03/82.45
Token-Wise Clipping	37.64	81.13/81.26	85.59/79.9	79.66	85.83/89.26	64.62	91.63	83.10/83.51
Gamma Migration + Token-Wise Clipping	**46.35**	**83.38/83.32**	87.50/83.33	**86.82**	**86.82/90.01**	**67.51**	**92.2**	**86.83/86.93**

Table 3: Results of our proposed Gamma Migration and Token-Wise Clipping for RoBERTa with 6-bit PTQ.

5.3 Main Results

5.3.1 Results on GLUE Benchmark

PTQ. Table 4 shows the results of PTQ on GLUE tasks. For 8-bit BERT models, although previous methods already behave well, our methods can still achieve satisfying outcomes even on small datasets such as CoLA (4.49% upswings) and STS-B (1.33%). To fully exploit the limit, we try more challenging settings with 6-bit weight and activation. It can be seen that ours is indeed close to FP value within 5.2% overall. Meanwhile, as PEG [13] additionally quantizes the "Add" operator and

Method	Bits (W-E-A)	Model Size (MB)	CoLA (Matt.)	MNLI (acc m/mm)	MRPC (f1/acc)	QNLI (acc)	QQP (f1/acc)	RTE (acc)	SST-2 (acc)	STS-B (Pear./Spear.)
BERT	32-32-32	417.6	59.60	84.94/84.76	91.35/87.75	91.84	87.82/90.91	72.56	93.35	89.70/89.28
MinMax	8-8-8	104.8	57.08	82.77/83.47	89.90/85.78	90.76	87.84/90.74	69.68	92.78	86.83/88.56
OMSE [15]	8-8-8	104.8	57.15	84.04/84.29	90.10/85.78	91.12	87.64/90.54	72.20	93.23	87.90/88.65
Ours	8-8-8	104.8	**61.64**	**84.38/84.53**	**91.44/87.75**	**91.49**	**87.92/90.77**	**72.20**	**93.81**	**89.23/89.01**
OMSE	6-6-6	78.7	35.44	74.00/73.30	81.54/76.47	84.66	76.07/82.12	64.26	86.27	85.57/86.05
Percentile [26]	6-6-6	78.7	37.32	72.40/71.69	85.09/79.90	79.37	72.58/80.19	61.73	87.27	86.38/87.29
EasyQuant [27]	6-6-6	78.7	44.18	78.27/79.24	85.12/77.21	81.97	72.15/79.71	62.45	85.44	82.19/82.03
Ours	6-6-6	78.7	**54.40**	**82.02/81.69**	**87.45/83.33**	**89.82**	**84.69/88.94**	**70.76**	**91.86**	**88.65/88.55**
PEG [13] ♣	8-8-8	104.8	59.43	81.25	88.53	**91.07**	**89.42**	69.31	92.66	87.92
Ours ♣	8-8-8	104.8	**59.83**	**82.93/82.59**	**91.33/87.99**	90.02	87.45/90.34	**70.04**	**92.66**	**88.42/88.81**
PEG ♣	6-6-6	78.7	9.46	32.44/32.77	83.64/78.43	49.46	29.93/62.97	**70.76**	90.14	52.79/53.22
Ours ♣	6-6-6	78.7	**42.27**	**78.54/78.32**	**85.33/81.13**	**85.36**	**78.47/84.66**	68.59	**91.74**	**87.33/87.19**
RoBERTa	32-32-32	475.5	62.50	87.75/87.23	93.1/90.44	92.68	88.78/91.6	80.51	95.18	91.04/90.72
MinMax	8-8-8	119.2	41.62	87.52/86.88	91.56/88.48	92.11	88.60/91.44	76.90	94.82	91.00/90.66
OMSE	8-8-8	119.2	38.59	87.32/87.14	92.39/89.46	92.51	87.95/90.95	76.53	94.61	90.95/90.65
Ours	8-8-8	119.2	**62.50**	**87.61/87.31**	**92.39/89.46**	**92.53**	**88.64/91.49**	**78.34**	**94.95**	**91.08/90.73**
OMSE	6-6-6	89.5	1.81	72.89/72.65	85.38/78.68	76.53	85.24/88.94	64.26	91.17	80.81/81.99
Percentile	6-6-6	89.5	20.73	72.23/73.68	84.83/78.43	77.16	82.21/87.44	62.82	88.19	79.41/79.64
EasyQuant	6-6-6	89.5	17.65	74.54/74.76	82.96/74.02	81.97	78.56/82.99	61.73	86.24	81.05/81.06
Ours	6-6-6	89.5	**46.35**	**83.38/83.32**	**87.50/83.33**	**86.82**	**86.82/90.01**	**67.51**	**92.2**	**86.83/86.93**
BART	32-32-32	534.1	56.32	86.45/86.55	91.37/87.50	92.31	88.34/91.39	79.06	93.35	90.11/89.94
MinMax	8-8-8	134.0	55.38	85.87/86.14	89.44/85.29	91.20	88.07/91.24	77.98	93.69	89.90/89.73
OMSE	8-8-8	134.0	54.56	85.6/86.25	90.31/86.27	90.74	88.21/91.3	78.7	93.58	90.07/89.88
Ours	8-8-8	134.0	**55.53**	**86.28/86.17**	**90.40/86.52**	**91.47**	**88.25/91.35**	**80.51**	**93.92**	**90.20/89.95**
OMSE	6-6-6	100.6	31.06	41.92/42.08	56.37/54.36	52.72	78.96/86.02	51.99	87.39	84.38/85.69
Percentile	6-6-6	100.6	26.21	74.72/75.29	83.52/74.26	53.71	82.64/87.48	67.15	87.96	63.99/65.01
EasyQuant	6-6-6	100.6	23.64	64.57/66.03	83.52/74.26	55.61	72.15/79.71	59.57	88.99	76.69/77.05
Ours	6-6-6	100.6	**44.51**	**82.46/82.98**	**86.41/80.88**	**86.34**	**83.60/88.45**	**71.12**	**90.94**	**87.56/87.38**

Table 4: PTQ performance on GLUE benchmark. ♣ indicates using quantization nodes of PEG [13] for thorough comparison. For the percentile , we search it in [0.999, 0.9999, 0.99999] and report the best on dev set.

uses per-layer weight quantization. For a fair comparison, we apply our mechanism to their setting. Favourable results on both 6-bit and 8-bit reveal the flexibility and the generality of our Gamma Migration and Token-Wise Clipping. To be noted, their per-embedding-group (PEG) quantization certainly brings extra computation and might not be available on real deployment.

Besides, the experimental results on RoBERTa and BART show that with 6-bit activation, the existing methods suffer from non-negligible accuracy drops, while ours consistently achieves satisfying results. To conclude, our proposed methods push the limit of 6-bit quantization to a new state of the art.

QAT. In particular, we measure the effectiveness of our methods on QAT with BERT model. Other models see Sec. D.4. In a much harder setting (4-4-4 bit quantization), our methods enable a good initialization to attain an acceptable accuracy drop (0.58% on QQP, 1.89% on MNLI) without any distillation and data argumentation trick, versus 3.67% and 3.51% on LSQ+. Furthermore, ours delivers improvements even when coupled with knowledge distillation, especially at 2-bit weight and embedding. To summarize, our outlier suppression framework achieves near-floating point performance with a reduction 2.40% on average on 4-bit quantization.

5.3.2 Results on SQuAD

To demonstrate the wider applicability of our methods, we evaluate them on question answering datasets. Most of the techniques work well for 8-bit. However, the performance drastically drops when going down to 6-bit quantization. Ours still outperforms others by a large margin. For example, our method improves 3.79% on BERT and 8.69% on RoBERTa.

5.3.3 Results on Summarization Tasks

It is of high value to investigate effect of our methods on generazation tasks. And we implement them on summarization tasks, which produces a brief statement of the article. We report the ROUGE 1/2/L score of BART on CNN/DailyMail and XSum. Table 7 illustrates that our approaches also benefit the Seq2Seq models, and can bring a near-floating point performance on 8-bit and about 4% enhancement on 6-bit.

Method	Bits (W-E-A)	BERT		RoBERTa		BART	
		SQuAD v1.1	SQuAD v2.0	SQuAD v1.1	SQuAD v2.0	SQuAD v1.1	SQuAD v2.0
Full Prec.	32-32-32	88.28/80.82	77.34/73.60	92.25/85.83	83.30/80.26	91.63/84.79	80.82/77.41
OMSE [15]	8-8-8	87.55/79.80	77.00/73.23	91.48/84.52	82.79/79.65	90.35/83.04	80.00/76.37
Ours	8-8-8	**87.96/80.32**	**77.30/73.57**	**91.60/84.95**	**82.92/79.80**	**90.47/82.95**	**80.45/76.91**
OMSE	6-6-6	80.33/69.43	68.77/64.07	70.70/59.09	45.52/39.69	81.30/70.69	67.92/63.18
Percentile [26]	6-6-6	78.73/67.58	68.60/64.62	67.85/55.07	56.13/51.18	80.70/70.80	72.70/67.81
EasyQuant [27]	6-6-6	80.92/70.87	69.77/65.11	65.38/51.18	44.04/37.32	78.53/67.24	61.37/56.42
Ours	6-6-6	**84.71/75.55**	**72.95/68.75**	**79.39/69.35**	**64.39/59.57**	**82.75/73.37**	**75.60/72.10**

Table 5: Comparison among typical PTQ approaches in terms of f1/em on SQuAD.

Method	Bits (W-E-A)	Model Size (MB)	CoLA (Matt.)	MNLI (acc m/mm)	MRPC (f1/acc)	QNLI (acc)	QQP (f1/acc)	RTE (acc)	SST-2 (acc)	STS-B (Pear./Spear.)
BERT	32-32-32	417.6	59.60	84.94/84.76	91.35/87.75	91.84	87.82/90.91	72.56	93.35	89.70/89.28
Q8BERT [6]	8-8-8	104.8	58.48	-	89.56/-	90.62	87.96/-	68.78	92.24	89.04/-
Q-BERT [7]	8-4-8	93.4	-	78.08/78.96	-	-	-	-	85.55	-
PACT [29]	4-4-8	52.6	55.23	83.98/83.9	91.58/88.24	91.12	88.19/91.2	71.84	91.86	89.73/89.27
LSQ+ [28]	4-4-8	52.6	57.7	84.17/84.02	89.75/85.78	91.27	88.18/91.16	70.76	91.97	**89.74/89.3**
PEG [13]	4-4-8	52.6	57.42	84.22/84.52	89.90/85.78	90.46	88.15/91.25	67.87	92.78	89.36/88.95
Ours	4-4-8	52.6	**61.06**	**84.82/84.89**	**91.26/87.75**	**91.41**	**88.45/91.4**	**73.65**	**92.55**	89.71/89.24
PEG	4-4-4	52.6	0.0	35.45/35.22	81.22/68.38	49.46	0.0/63.18	52.71	76.26	nan/nan
PACT	4-4-4	52.6	0.0	74.17/74.85	84.97/80.15	87.31	81.68/86.14	62.09	83.03	81.64/81.43
LSQ+	4-4-4	52.6	0.0	81.4/81.97	88.34/83.82	88.1	83.11/87.24	64.62	82.34	84.16/83.75
Ours	4-4-4	52.6	**50.56**	**83.05/83.24**	**89.08/84.31**	**89.88**	**87.00/90.33**	**70.76**	**91.86**	**87.64/87.36**
PEG ♣ *	4-4-8	52.6	57.22	83.69	87.77	91.29	89.64	70.04	92.32	89.13
Ours ♣	4-4-8	52.6	**59.57**	**85/84.31**	**91.07/87.75**	**91.31**	**88.35/91.32**	**72.2**	**92.43**	**89.57/89.2**
PEG ♣	4-4-4	52.6	0.0	35.45/35.22	31.62/0.0	49.46	0.0/63.18	52.71	49.08	-0.0219/-0.0199
Ours ♣	4-4-4	52.6	**51.93**	**83.03/83.24**	**89.39/85.05**	**90.33**	**87.38/90.62**	**72.56**	**91.74**	**88.36/87.91**
LSQ+(+KD)	4-4-4	52.6	12.72	83.73/83.82	90.4/86.52	90.61	87.17/90.47	66.06	84.4	84.23/84.01
Ours(+KD)	4-4-4	52.6	**56.1**	**84.67/85.06**	**91.26/87.75**	**91.45**	**88.69/91.56**	**72.2**	**92.89**	**88.49/88.14**
LSQ+(+KD)	2-2-4	26.5	0.3	82.18/82.74	89.23/84.56	89.97	86.33/89.33	56.68	84.63	38.81/38.92
Ours(+KD)	2-2-4	26.5	**44.18**	**83.84/83.95**	**89.42/84.8**	**90.55**	**88.21/91.17**	**63.18**	**91.97**	**83.01/82.95**

Table 6: Comparison among different QAT strategies with low-bit activation on GLUE benchmark for BERT. ♣Uses the same quantization nodes as PEG [13] for thorough comparison. *Reports the combined score for MNLI, MRPC, QQP and STS-B, which is the average of the metrics.

Method	Bits(W-E-A)	CNN DailyMail	XSum	Bits(W-E-A)	CNN DailyMail	XSum
Full prec.	32-32-32	45.62/22.85/42.88	42.82/20.11/34.99	32-32-32	45.62/22.85/42.88	42.82/20.11/34.99
OMSE [15]	8-8-8	44.89/22.03/42.18	41.58/18.77/33.73	6-6-6	37.56/15.46/34.92	16.11/2.13/12.22
Percentile [26]	8-8-8	44.67/21.74/41.81	41.47/18.67/33.61	6-6-6	37.02/15.31/34.45	30.10/9.43/22.70
EasyQuant [27]	8-8-8	44.32/21.37/41.53	41.40/18.74/33.53	6-6-6	36.71/14.99/34.49	21.72/6.28/16.87
Ours	8-8-8	**45.70/22.89/43.15**	**42.32/19.74/34.59**	6-6-6	**40.12/17.62/37.24**	**34.50/12.69/26.10**

Table 7: PTQ results of BART model on summarization tasks. The model size of different bit settings: FP: 531.8MB, 8-bit: 133.4MB, 6-bit: 100.2MB.

6 Conclusions and Discussions of Limitations

In this paper, we analyze the outlier phenomenon from the inducement and clipping impact aspects on language transformer models. Then we establish an outlier suppression framework to combat the quantization challenges and comprehensively verify the effectiveness on a large variety of tasks. There also remain some open problems worthy of more in-depth investigations. For example, it is valuable to systematically explore whether the conclusion in this paper benefits other fields such as computer vision. Besides, as we supplement in the Appendix that the outliers occur not only in the fine-tuned (BERT) models but also the pre-trained ones, it's also meaningful to dive into the pre-training process for a better understanding.

References

[1] 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.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[4] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[5] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[6] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE, 2019.

[7] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821, 2020.

[8] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*, 2020.

[9] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*, 2020.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR, 2021.

[12] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020.

[13] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv preprint arXiv:2109.12948*, 2021.

[14] Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier layernorm dimensions that disrupt bert. *arXiv e-prints*, pages arXiv–2105, 2021.

[15] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019.

[16] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[17] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[18] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[19] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[20] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[21] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[22] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789. Association for Computational Linguistics, 2018.

[23] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics, 2018.

[24] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics, 2016.

[25] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[26] Jeffrey L. McKinstry, Steven K. Esser, Rathinakumar Appuswamy, Deepika Bablani, John V. Arthur, Izzet B. Yildiz, and Dharmendra S. Modha. Discovering low-precision networks close to full-precision networks for efficient embedded inference, 2019.

[27] Di Wu, Qi Tang, Yongle Zhao, Ming Zhang, Ying Fu, and Debing Zhang. Easyquant: Post-training quantization via scale optimization. *arXiv preprint arXiv:2006.16669*, 2020.

[28] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 696–697, 2020.

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

[30] FasterTransformer. Fastertransformer. https://github.com/NVIDIA/FasterTransformer.

[31] NNCF. Nncf. https://github.com/openvinotoolkit/nncf/tree/develop/third_party_integration/huggingface_transformers.

[32] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

[33] Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Discussions we leave some topics as future work.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Detailed proofs can be found in the supplementary materials.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide code of experiment as part of our supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We defer detailed training settings in the supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Since we comprehensively evaluate the robust generalization for various models on different datasets, it would be computationally expensive to have the error bar.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]