# 1. Analysis:

## 1.1 Why does the second term in Eq.4 disappear?

The process of model training involves minimizing the loss function with respect to the weights (So, we use the optimization methods such as SGD, ADAM, etc. to train model). After the training process, the loss function reaches its minimum value, i.e., $w = \underset{w}{\operatorname{argmin}} f(w, dataset)$. Furthermore, according to the properties of extrema, the gradient of the weights is 0. Moreover, in experiments, we have also observed gradients of weight smaller than 1e-7, which can be ignored. Therefore, for a well-trained model, the derivative of the loss function with respect to the weights is 0, leading to Eq.4.

## 1.2 Why we do not use the second order elements information to deal quantization problems. The authors are ignoring the second order elements, which is frequently discussed in works such as BRECQ, PTQ4ViT etc.

From a theoretical analysis perspective, regarding the use of second-order terms in other works, we find that after analyzing BRECQ or PTQ4VIT, in these works, they only Taylor-expand and analyze the weights because the gradient of the loss with respect to the weights' first-order terms is close to 0, so only second-order terms are used for analysis. However, BRECQ and PTQ4VIT lack theoretical analysis on activations and avoid the analysis of influence about noise introduced in activations on the loss function, which contradicts the principles of quantization.

From the experimental perspective, when analyzing the impact of noise introduced in activations on the loss function, the effect of second-order terms on weights and activations is generally smaller than O(2.5e-5), and the impact on final accuracy is relatively minor (fluctuations in Top1 accuracy are around 0.01, and in Top5 accuracy are around 0.001). This indicates that the first-order terms are the primary factors causing changes because the influence of second-order terms is infinitesimal higher-order terms. The experimental results are consistent with theoretical analysis.

In fact, works like BRECQ seem to have clear analysis of quantization algorithms, but during algorithm implementation, they require a fine-tuning process, which is not

included in the corresponding theoretical analysis, leading to a disconnection between theoretical analysis and algorithm design. After fine-tuning, the entire parameters undergo significant changes, rendering local analysis using Taylor expansion ineffective. Therefore, these works rely on fine-tuning operations that are not explicitly mentioned in the analysis. Moreover, through experiments, we found that removing the fine-tuning process renders these works ineffective (drastic decrease in accuracy). Conversely, our work remains consistent with theory even without fine-tuning, further demonstrating the validity and rationality of our theoretical analysis.

Furthermore, such works should be used in scenarios where models are stored on disk in low precision but converted to high precision during computation and storage in memory. Whereas our work utilizes low precision throughout the computation process.

### 1.3 Why the difference in the final loss value can be decomposed by the loss of each layer? why ϵ vector require differential calculation?

As mentioned in the main paper, we treat neural networks as composite functions and conduct theoretical analysis using total differentials. In total differentials, for any differentiable function $f(w_i, w_j)$, with small variables $\delta_i$ and $\delta_j$, we have $f(w_i+\delta_i, w_j+\delta_j) - f(w_i, w_j) = \frac{\partial f}{\partial w_i}\delta_i + \frac{\partial f}{\partial w_j}\delta_j$. If $w_i$ and $w_j$ are vectors, the above equation still holds [1]. This is the general form of total differential expansion. We regard the loss function as the composite function $f$ in the above formula, where the $w_i$ vector is considered as the weights or activations of one layer, and the $w_j$ vector is considered as the weights or activations of another layer. Thus, we obtain Eq.3 in the paper.

### 1.4 Why use secant line?

The purpose of Figure 3 is to illustrate the use case of secant lines. The use of tangent lines is limited and depends on the size of the neighborhood in mathematics.

However, this limitation does not apply to secant lines because their definition is domain-independent. Tangent lines are the limit of secant lines, so tangent lines can only be used effectively within a domain where the magnitude of noise falls within the mathematical concept of the neighborhood .

When using the algorithm described in the paper, we have unified the scenarios for using secant and tangent methods. When $\epsilon$ is small enough, meaning the noise magnitude falls within the mathematical domain, the result obtained using secant lines is equivalent to the result obtained using the tangent method because the tangent method is the limit of the secant method.

**1.5 It should be some simple description or citation for 'convenient language of probability theory'.**

To ensure the readability of the paper, the current paper version employs convenient language in probability theory. It provides the easiest analysis method. If reviewer want to see more explanation, there are also other analytical: Analysis 1 and Analysis 2. Analysis 1 and 2 require more complex mathematical skill or more verbose descriptions, but the conclusions remain unchanged.

For the noise vector e, it must be a vector of independent and identically distributed (i.i.d.) random variables. This is because the noise vector essentially quantifies the residual (similar to determining the distribution of the decimal part of a number). There are different interpretations of how the $\frac{\partial \ell}{\partial h_{i+1}}$ vector, $[p_1, p_2, ..., p_k]$ should be understood. In the paper, the vector $\frac{\partial \ell}{\partial h_{i+1}}$ is interpreted as a vector of random variables, with each random variable being i.i.d. This method simplifies the derivation of Eq. 6 and $\mu = |\frac{\partial \ell}{\partial h_{i+1}} \cdot \vec{1}|$. In fact, it is also acceptable to interpret the vector $\frac{\partial \ell}{\partial h_{i+1}}$ as a whole random variable vector, where each random variable element is not i.i.d., or simply as a fixed vector determined by the training set.

Analysis 1: Interpreting the vector $\frac{\partial \ell}{\partial h_{i+1}}$ as a whole random variable vector, where each random variable element is not i.i.d.

$$\mathbf{E}\frac{\partial \ell}{\partial h_{i+1}} \cdot \epsilon = \mathbf{E}\sum_{i=1}^{k} e_i * p_i = \mathbf{E}e(\sum_{i=1}^{k} \mathbf{E}p_i) = \mathbf{E}e\mathbf{E}(\sum_{i=1}^{k} p_i)$$

Constructing a new random variable q, $q = \sum_{i=1}^{k} p_i$ , then $\mathbf{E}\frac{\partial \ell}{\partial h_{i+1}} \cdot \epsilon = \mathbf{E}e\mathbf{E}q$ . By definition, $\mathbf{E}q = |\frac{\partial \ell}{\partial h_{i+1}} \cdot \vec{1}|$.

Based on the above analysis and the Chebyshev's inequality, we can infer

$$P(\frac{\partial \ell}{\partial h_{i+1}} \cdot \epsilon \geq 0) < P(|\frac{\partial \ell}{\partial h_{i+1}} \cdot \epsilon - \mathbf{E}e\mathbf{E}q| \geq |\mathbf{E}e\mathbf{E}q|)$$

$$\leq \frac{Var(eq)}{|\mathbf{E}e\mathbf{E}q|^2} = \frac{Var(e)Var(q)}{|\mathbf{E}e\mathbf{E}q|^2} + \frac{Var(e)}{|\mathbf{E}e|^2} + \frac{Var(q)}{|\mathbf{E}q|^2}$$

Hence, when $\mathbf{E}q$ is larger, i.e., $|\frac{\partial \ell}{\partial h_{i+1}} \cdot \vec{1}|$ is larger, $Var(q)$ is smaller, making it more likely to obtain good results.

Analysis 2: Interpreting the vector $\frac{\partial \ell}{\partial h_{i+1}}$ as a fixed vector determined by the training set.

$\mathbf{E}\frac{\partial \ell}{\partial h_{i+1}} \cdot \epsilon = \mathbf{E}\sum_{i=1}^{k} e_i * p_i = \mathbf{E}e(\sum_{i=1}^{k} p_i)$ where $\sum_{i=1}^{k} p_i = \frac{\partial \ell}{\partial h_{i+1}} \cdot \vec{1}$ . When the training set, validation set, and test set change significantly, the sign obtained on the validation set may change, leading to failure. Also, when $\sum_{i=1}^{k} p_i$ is close to 0, even tiny perturbations can change its sign, also leading to failure. Therefore, the larger $\sum_{i=1}^{k} p_i$ is, and the smaller the numerical impact of the training set, test set, and validation set on it, the easier the algorithm is to succeed.

# 2. Experiments:

## 2.1 Comparison Experiment

ResNet18 comparison experiment

| Resnet18 | Acc@1 | Acc@5 | losses | w/o finetuning |
|---|---|---|---|---|
| orgin | 69.77 | 89.07 | 1.247004 | |
| HAWQ | 69.556 | 88.966 | 1.254416 | w/ |
| AdaRound | 68.55 | - | - | w/ |
| ACIQ | 69.628 | 89.006 | 1.249251 | w/o |
| ours | 69.75±0.005 | 89.092±0.006 | **1.246837±0.00002** | w/o |

Mobilenet_v2 comparison experiment

| mobilenet_v2 | Acc@1 | Acc@5 | losses | w/o finetuning |
|---|---|---|---|---|
| orgin | 71.89 | 90.292 | 1.14792 | |
| HAWQ | **72.906** | **90.974** | 1.17033 | w/ |
| AdaRound | 69.25 | - | - | w/ |
| HAQ | 71.85 | 90.24 | - | w/ |
| BRECQ | 72.57 | 90.672 | 1.1956 | w/ |
| ours | 71.882±0.002 | 90.298±0.003 | **1.147831±0.000001** | w/o |

Bert comparison experiment

| Bert | EM | F1 |
|---|---|---|
| orgin | 80.492 | **88.1464** |
| FP8-BERT(AAAI 2023) | 78.89 | 86.44 |
| ours | **80.5109** | 88.1458 |

Model size comparison experiments

| Model size(MB) | resnet18 | mobilenet_v2 | Bert |
|---|---|---|---|
| orgin | 44.7 | 13.6 | 421 |
| ours | 12.27 | 4.13 | 254.25 |
| HAWQ | 11.1 | - | - |

The above table shows our comparative experiment. Comparative experiments show that our method outperforms other methods. It is worth noting that most existing methods require fine-tuning to obtain better results. Our method has lower losses than existing methods without any fine-tuning process, and is also competitive in accuracy. At the same time, our method can quantize the model, making the size of the quantized model smaller and reducing the loss. This shows that our method quantizes the model

losslessly.

In terms of running time, our quantitative method is very fast. Taking resnet18 as an example, the overall time is less than 5 minutes. And our method does not require fine-tuning, and the fine-tuning time is 0. Our method is lossless quantization and has fast running speed, which has significant advantages.

For Resnet18 as an example, we quantized all layers except the feature extraction layer. For quantization layers in Bert, we quantize all layers except embedding and the second layer. For other models, we will explain in detail in the article later. Please refer to the experimental section to quantify model size.

Due to the limitation of word count and time in reply, we will put all relevant comparative experiments into the main paper in the future.

**2.2 Ablation Study**

The ablation experiment about $\mu$

| Model+ $\mu$ | Loss | Acc | Data | Type |
|---|---|---|---|---|
| CNN_R+0.3 | 0.1167 | 0.9211 | MNIST | I4,I8,F |
| CNN_B+0.3 | 0.0654 | 0.9731 | MNIST | I4,I8,F |
| CNN_R+0.5 | 0.0782 | 0.9748 | MNIST | I4,I8,F |
| CNN_B+0.5 | 0.0782 | 0.9748 | MNIST | I4,I8,F |
| CNN_R+0.4 | 0.0801 | 0.9691 | MNIST | I4,I8,F |
| CNN_B+0.4 | 0.0654 | 0.9731 | MNIST | I4,I8,F |

In the ablation experiment above of the $\mu$ experiment in dataset MNIST, when the value of $\mu$ is too small, the RMILS algorithm is radical and quantifies some of the more sensitive layers in the CNN. Because there are differences between the calibration set and the test set, the CNN_B algorithm Performance is worse than full precision. When the value of $\mu$ is large, although most layers can help the model and further improve the effect through quantization, they cannot be quantified, resulting in limited loss reduction.

For $\sigma$, it only represents the variance of the activation gradient. It only appears in

the theoretical analysis, but does not appear in the algorithm, so the current algorithm has nothing to do with $\sigma$.

## Reference

[1] Matthews P C. Vector calculus[M]. Springer Science & Business Media, 2012.