BitNet and Quantization-Aware Training

Aiwei Yin[1] and Qingyang Bao[1]

[1]Univeristy of Toronto

January 8, 2025

1 Introduction

As a way to drastically decrease computing cost while maintaining similar performance, quantization of large models has gained considerable interest. However, most work keep presision after quantization more than 4 bits. The work (Wang et al., 2023) introduces 1-bit quantization, reducing the weight to only $\{-1, 1\}$ (and in the follow-up work, $\{-1, 0, 1\}$ (Ma et al., 2024)) while maintaining similar performance experimented on Large Language Models (LLMs) of various sizes.

2 Background

Quantization involves decreasing the bit precision of a model's parameters, aiming to minimize any impact on inference performance. There are two main types of quantization: Quantization-Aware Training (QAT) and Post-Training Quantization (Gholami et al., 2021). The key difference between them is whether retraining is necessary. PTQ allows the quantized model to be used directly for inference, whereas QAT requires retraining to rectify errors introduced by quantization.

In this work, we mainly discuss Quantization-Aware Training and focus on the extremely low bit-width deployment of LLMs.

3 BitNet

In this work, they Wang et al., 2023 focus on binarization, which is the extreme case of quantization, applied to large language models. There are some previous works on binarized neural network in the era before the rise of large language models.

So they propose a 1-bit Transformer architecture for large language models. `BitNet`, a LLAMA-like (Grattafiori et al., 2024) Decoder-Only Transformer, but use `BitLinear` instead of conventional linear layers in the feed-forward networks (FFNs) as well as QKV projection matrices, while attention layers unchanged. The reasoning for only applying quantization on the FFNs is that they take up most of the computation as the model scale up, so quantizing linear layers can get most significant decrease in computation cost. The customized `BitLinear` layer is specialized for quantization.

3.1 Quantization Method

In a `BitLinear` layer, the weight matrix $W \in \mathbb{R}^{n \times m}$ is binarized into $\{-1, 1\}$-valued, by applying binarization. (Wang et al., 2023) It is worth noting that after such quantization, since all entries of the weight matrix is either -1 or 1, the matrix multiplication only consists of addition and subtraction operations.

$$\widetilde{W} = \text{Sign}(W - \alpha) \tag{1}$$

$$\text{Sign}(W_{ij}) = \begin{cases} +1, & \text{if } W_{ij} > 0 \\ -1, & \text{if } W_{ij} \leq 0 \end{cases} \tag{2}$$

$$\alpha = \frac{1}{nm} \sum_{ij} W_{ij} \tag{3}$$

The activation x is also quantized to b-bit precision using AbsMax quantization. (Dettmers et al., 2022)

$$\widetilde{x} = \text{Quant}(x) = \text{Clip}(x \times \frac{Q_b}{\gamma}, -\frac{Q_b}{\gamma} + \epsilon, \frac{Q_b}{\gamma} - \epsilon), \tag{4}$$

$$\text{Clip}(x, a, b) = \max(a, \min(b, x)),\ \gamma = ||x||_\infty \tag{5}$$

Along with the above quantization, `BitLinear` also introduces several ways to account for the change in variance after quantization. To keep the variance be approximately 1 (as most full-precision models with Kaiming Initialization do), a layer normalization is added to ensure $\text{Var}(\widetilde{x}) = 1$. Let $y = \widetilde{W}\widetilde{x}$, and assume $\widetilde{W}$ and $\widetilde{x}$ are mutual independent, sharing the same distribution, we have

$$\begin{aligned} \text{Var}(\widetilde{W\widetilde{x}}) &= n\text{Var}(\widetilde{w}\widetilde{x}) & (6) \\ &= nE[\widetilde{w}^2]E[\widetilde{x}^2] & (7) \\ &= n\beta^2 E[\widetilde{x^2}] & (8) \end{aligned}$$

To align with unquantized models, where $Var[y] = 1$, a layer norm is applied before matrix multiplication.

3.2 Model Training

During training the model keeps a latent weights with full precision, and apply quantization on the latent weights and activations during the forward step.

Straight-through estimator (Bengio et al., 2013) Since the process of weight and activation quantization, in particular, the Sign and Clip functions, is not differentiable, straght-through gradient estimator is used to bypass the non-differentiable steps, so that gradient can be propagated back. This method provides an easy way for the model to back propagate, but does not necessarily captures the accurate error to propagate.

Let x be a value on the forward pass. x and $\widetilde{x}$ are assumed to be very similar. During the forward pass, $\widetilde{x}$ is replaced with $x + sg[\widetilde{x} - x]$, where sg is the stop gradient operator (`detach()` method in PyTorch) (cite).

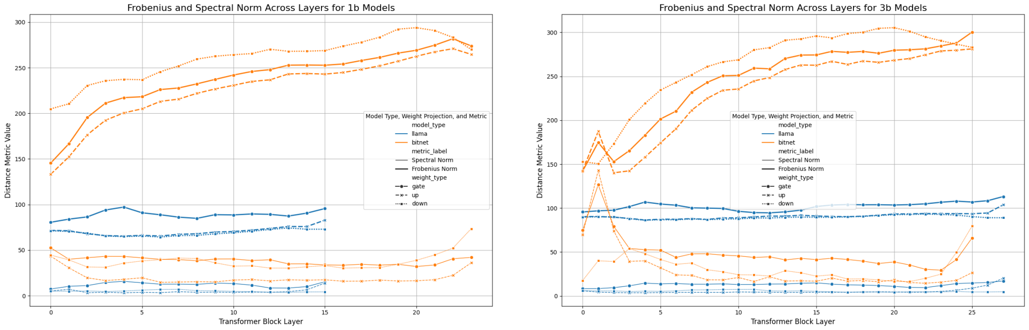
By the chain rule, we have

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial \widetilde{x}} \frac{\partial \widetilde{x}}{\partial x}$$

where $\mathcal{L}$ is the loss. While $\frac{\partial \widetilde{x}}{\partial x}$ is not well-defined, the STE assumes $\widetilde{x} \approx x$, hence $\frac{\partial \widetilde{x}}{\partial x} \approx I$, I being the identity matrix. As a result the gradient will be back-propagated to x, in turns effects $\widetilde{x}$ after GD update.

It is worth noting that BitNet's use of STE's is special in that it stacks a large number of STE's across all the transformer blocks. This is unlike previous models such as VQ-VAE, that utilize only one layer of STE.

Large Learning Rate Due to the aggresive binarization, a small update on the latent weight will very likely have no effect on the binarized weight, especially at the beginning of training. After experiments, the authors decided to increase the learning rate to overcome this issue. Experiments have also shown that BitNet can stay stable at large learning rate, that full-precision transformers fail to converge. (Wang et al., 2023)



(a) Frobenius Norm of 3B models

(b) Frobenius Norm of 1B models

Figure 1: Frobenius norm of models

4 Analysis on BitNet and QAT

The original paper evaluate BitNet on a range of benchmarks, comparing with state-of-the-art quantization methods and FP16 Transformers. Moreover, BitNet substantially lowers memory usage and energy consumption in comparison to the baseline models.
More importantly, they have shown that BitNet follows a scaling law similar to transformer. It shows that Bit quantization can be scaled up for future large language models with much faster inference speed and similar efficiency.

4.1 Experiments

Our experiment focus on analyzing the weight matrices of the BitLinear layers to study what effect of the training methods mentioned above would have on the weight matrices. We choose a open-source reporduction of the follow-up paper (Ma et al., 2024), and Meta's LLAMA model (Grattafiori et al., 2024). The follow-up paper only slightly changes the quantization method, and uses a very similar architecture to the well-known LLAMA model, which provides us an ideal environment to compare and contrast. From both sides, we choose two models of 1B and 3B, which is 4 models in total. We compare the model with same parameter size, to minimize the effect of scaling on model weights. We analyze norm of weight matrices on both the models. For the norm metric, we use Frobenius norm and spectral norm. Frobenius norm treats matrices as a vector of all its entries, and is similar to the Euclidean norm of vectors, providing an overall magnitude considering all entries. Spectral norm represents the maximum amplification it can have on an input vector, and can capture more accurate characteristics of the transformation.

We then analyze the similarity of each of the model's weight before and after quantization. We use Frobenius distance, spectral distance, and cosine similarity as metrics.

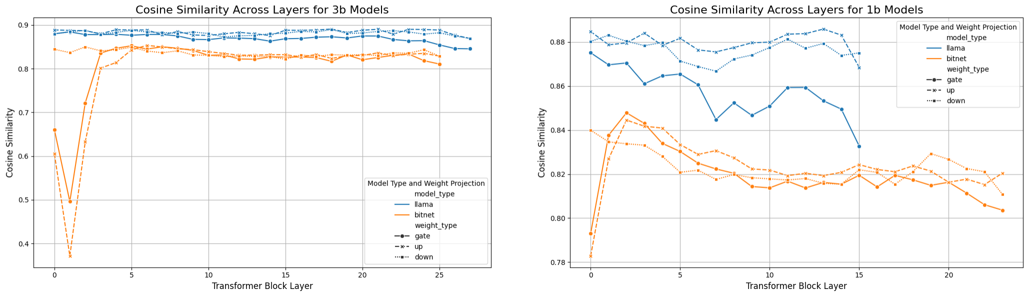
$$sim(W, \widetilde{W}) = \langle W, \widetilde{W} \rangle_F = tr(W \widetilde{W}^T)$$

$$dist_F(W, \widetilde{W}) = ||W - \widetilde{W}||_F, \; dist_S(W, \widetilde{W}) = ||W - \widetilde{W}||_S$$

4.2 Analysis on the result

As shown in 5, Our first finding is that the Frobenius norm of BitNet weights are significantly larger than LLAMA weights. This indicates that the entries in BitNet's weight matrices are significantly larger. Since BitNet's initialization process is not altered, we think that this is caused by the larger learning rate comapred to conventional models, which push the weights to a subspace with much further from the origin.

Gradient Vanish More importantly, 5 shows that the Frobenius norm of the layers shows a drastically decreasing trend as the layer depth decreases. In the 3B model, the Frobenius norm has decreased to only $\frac{1}{2}$, while the Frobenius norm of LLAMA models stay relatively constant across layers. We suspect that this is



(a) Cosine similarity of 3B models before/after quantization

(b) Cosine similarity of 1B models before/after quantization

Figure 2: Cosine similarity of models before/after quantization

cause by the gradient's decreasing in Frobenius norm, which indicates that stacking too many layers of STEs will have a significant gradient vanish. The trend is also more significant in 3B models compared to the 1B models. We therefore question on the ability of BitNet's architecture to scale to even larger parameter size, as that will inevitably stack more layers of STEs, leading to even more severe gradient vanish, and harder training.

The spectral norm, on the other hand, is relatively steady along the layers. They do not show a significant trend of increase or decrease, and is closer to the spectral norm of LLAMA models. However, the spectral norm is high on both the left-most and right-most layers in the two BitNet models.

Similarity of Matrices All of the similarities we analyzed does not have a significant sign that BitNet weight's loss after quantization is smaller. Both the Frobenius distance and spectral distance show a lower similarity after quantization, but is highly effected by the fact that BitNet weight matrices have a higher norm than LLAMA weight matrices. As for the scale-invariant cosine similarity, 3B BitNet model shows a comparable, but still lower cosine similarity of roughly 0.85 compared to roughly 0.88 on the 3B LLAMA model. We conclude that there does not seem to have a correlation between quantization loss and performance after quantization.

5 Limitations

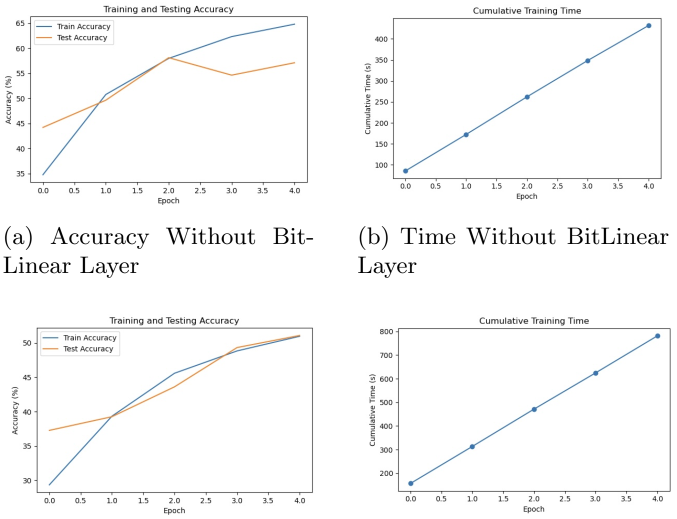
Quantization-Aware Training (QAT) effectively enhances the performance of quantized models by simulating quantization effects during training. However, it has notable drawbacks: it increases training complexity and cost due to longer training times and higher resource consumption. Implementing QAT is also more challenging than Post-Training Quantization (PTQ), requiring a deep understanding of the process and additional effort for debugging and optimization.

To address this issue, we conducted an experiment on a classic CNN structure. We replaced the original MLP layer with a Bitlinear layer and compared the results with those of the original full-precision model.

Based on our results 3 on the CIFAR-10 dataset, we found that the quantized model achieved almost the same accuracy after training for the same number of epochs but required twice the time compared to the original model. If the quantized model also follows a scaling law, the training process would become too time-consuming, making it impractical to incorporate this structure into state-of-the-art LLMs.

6 Future Work

Our result show that although BitNet model have a comparable performance as conventional models, the weights lies in a compeletly different space compared to conventional models without QAT, where norm of the weight is significantly larger. In other words, QAT can lead to another local minimum in the optimization landscape such that the performance loss after quantization is reduced. However, there is significant gradient



(a) Accuracy Without BitLinear Layer

(b) Time Without BitLinear Layer

(c) Accuracy With BitLinear Layer

(d) Time With BitLinear Layer

Figure 3: Training accuracy and time on ResNet with/without BitLinear.

vanish in the current QAT training methods, which can make training even harder when the model scales up, and has more depth.

In this study, we hypothesize that certain modifications to the Quantization-Aware Training (QAT) methodology may improve effectiveness. For instance, adjustments could be made to the Straight-Through Estimator (STE) that reduces gradient vanishing issues, and enhances training stability. Alternatively, bypassing QAT and exploring other approaches to achieve comparable local minima, such as new initialization techniques or learning rate strategies, could also be viable directions for optimization.

References

Bengio, Yoshua, Léonard, Nicholas, and Courville, Aaron (2013). *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*, arXiv: `1308.3432 [cs.LG]`. **available at**: `https://arxiv.org/abs/1308.3432`.

Dettmers, Tim et al. (2022). *LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale*, arXiv: `2208.07339 [cs.LG]`. **available at**: `https://arxiv.org/abs/2208.07339`.

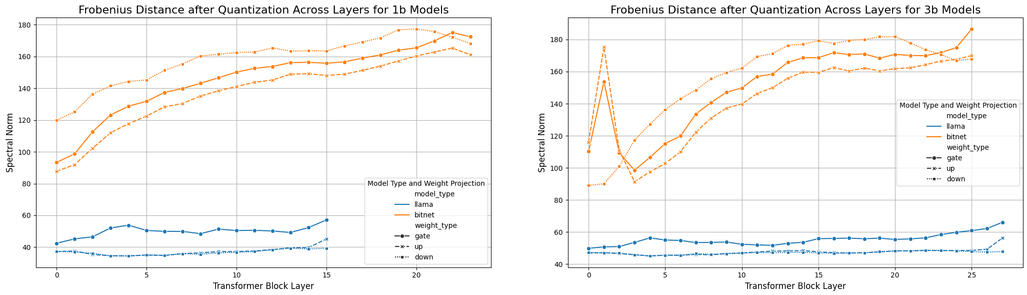
Gholami, Amir et al. (2021). *A Survey of Quantization Methods for Efficient Neural Network Inference*, arXiv: `2103.13630 [cs.CV]`. **available at**: `https://arxiv.org/abs/2103.13630`.

Grattafiori, Aaron et al. (2024). *The Llama 3 Herd of Models*, arXiv: `2407.21783 [cs.AI]`. **available at**: `https://arxiv.org/abs/2407.21783`.

Ma, Shuming et al. (2024). *The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits*, arXiv: `2402.17764 [cs.CL]`. **available at**: `https://arxiv.org/abs/2402.17764`.

Wang, Hongyu et al. (2023). *BitNet: Scaling 1-bit Transformers for Large Language Models*, arXiv: `2310.11453 [cs.CL]`. **available at**: `https://arxiv.org/abs/2310.11453`.

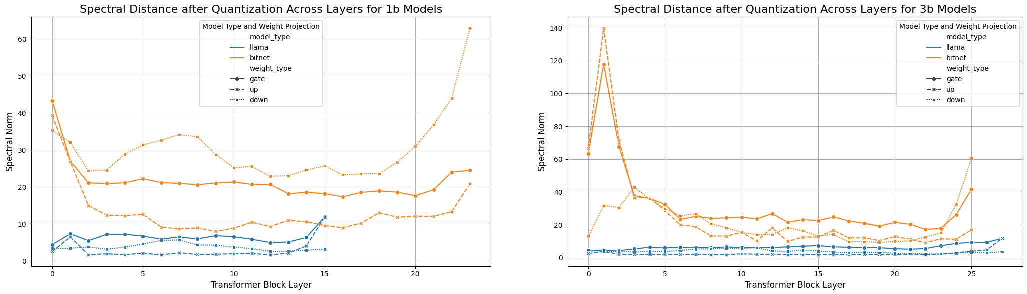
7 Appendix



(a) Frobenius distance before and after quantization of 1B models

(b) Frobenius distance before and after quantization of 3B models

Figure 4: Frobenius distance before and after quantization of models



(a) Spectral distance before and after quantization of 1B models

(b) Spectral distance before and after quantization of 3B models

Figure 5: Spectral distance before and after quantization of models