Reducing Training Time and Enhancing Efficiency in Large Language Models: Ternary Output Quantization(TOQ)

Do-hoon Lee

ABSTRACT

This study examines the background of recent advancements in artificial intelligence technology and the need for large-scale language models (LLM) and multimodal models (LMM), proposing the Ternary Output Quantization (TOQ) technique to reduce computational complexity and shorten training times for these models. TOQ simplifies the final outputs of models to three discrete values (-1, 0, +1), significantly reducing data processing requirements and enhancing the efficiency of model training and inference processes. Experimental results show that applying TOQ to BERT and DistilBERT models significantly reduces training times and improves both training and validation loss compared to traditional models. This paper presents the potential for wide application of TOQ across various forms of AI models, anticipating that it will facilitate easier training and deployment of models, especially in resource-constrained environments.

☞ keyword : AI, Ternary Output Quantization, LLM, Learning Efficiency, Data Processing Minimization

1. Introduction

The field of artificial intelligence has seen rapid advancements since the introduction of the Transformer model (Vaswani, et al., 2017). In particular, large language models (LLMs) and large multimodal models (LMMs), built on top of Transformers, have emerged, solving many previously intractable problems. These models, however, require massive computational resources to achieve their impressive performance (Vaswani et al., 2017; Brown et al., 2020; Kaplan et al., 2020; Narayanan, et al., 2021).

Large-scale models typically involve billions of paramet ers for training and inference, necessitating high-performan ce hardware, which implies high associated costs. These ch allenges become especially critical in resource-constrained environments. To address these issues, techniques such as Low-Rank Adaptation (LoRA) and Retrieval-Augmented G eneration (RAG) have been proposed. These methods aim to reduce the training burden by modifying only parts of the model or integrating external knowledge sources (Hu, et al., 2021; Lewis, et al., 2020). However, these approach es still struggle with the time and resource demands associ ated with large-scale data processing. In cases where suffic ient data and computational resources are available, fully tr aining a large model from scratch often achieves better per formance for specific tasks (Narayanan, et al., 2021).

This study hypothesizes that reducing the computational complexity by converting the complex and diverse continuous output variables of the model into three discrete values (-1, 0, +1) can improve model performance while maintaining similar accuracy across epochs. To this end, we propose a novel technique called Ternary Output Quantization (TOQ), which aims to reduce computational overhead by quantizing the model's output. TOQ simplifies the output to three distinct values (-1, 0, +1), thereby improving processing speed while maintaining or enhancing model performance. The introduction of TOQ is particularly beneficial in terms of energy efficiency and processing speed, making AI models more feasible for use in environments with limited resources.

The objective of this research is to reduce the computational cost involved in training and inference of large language models through the application of TOQ. Specifically, we aim to experimentally validate how TOQ enhances model training efficiency and inference speed, and evaluate its impact on generalization and overall performance. We also explore the applicability of TOQ by integrating it with models like BERT and DistilBERT to assess its utility in real-world natural language processing (NLP) tasks. The findings are expected to contribute significantly to improving model performance and reducing training time in NLP models, particularly facilitating the training and deployment of large-scale models in resource-limited environments.

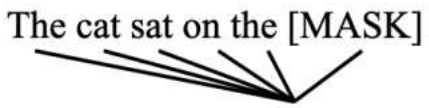
2. Theoretical Background and Related Work

2.1 Theoretical Background

2.1.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language representation model widely applied in natural language processing. It employs two main pre-training techniques, namely Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), to learn deep bidirectional representations.

Masked Language Model (MLM) is one of BERT's core pre-training mechanisms, where parts of the text are randomly masked, and the model is trained to predict these masked tokens based on the context. During this process, the model learns the ability to infer masked words from their surrounding context. For instance, in the sentence “The cat sat on the [MASK],” the model must predict the masked word using clues from “cat,” “sat,” “on,” and “the.” This mechanism helps BERT understand the bidirectional context of text. By predicting masked tokens, the model gains a deep understanding of the context, thereby learning richer language representations.



(Image 1) MLM

Next Sentence Prediction (NSP) is another important pre-training mechanism in BERT, focusing on determining whether two given sentences are sequentially related. For example, given the sentences “I drank coffee. [SEP] It was very strong,” BERT must decide whether these sentences logically follow one another. In contrast, given the sentences “I drank coffee. [SEP] She traveled to America,” BERT should recognize that the sentences are unrelated. Through NSP, BERT learns logical continuity and semantic connections between sentences, enhancing its understanding of overall document structure.

$$Attention(Q,\ K,\ V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

According to Equation (1), the attention mechanism calculates the relationships between input tokens using Query (Q), Key (K), and Value (V). This helps determine which tokens in the sentence are most related to each other (where $d_k$represents the dimension of the key vector). Q represents the question, K represents the target of that question, and Q matched with K returns the value V. Essentially, the degree of similarity between Q and K is represented as a probability, which is then multiplied by V to generate the final attention score. This information is used to generate new token representations.

BERT takes tokenized text as input and converts each token into embedding vectors. These vectors are combined with positional encodings to include information about the order of tokens in the sentence. Then, the input passes through multiple layers of Transformer encoders, which use multi-head self-attention to learn how each token relates to others in the sequence.

The training of BERT involves pre-training on large amounts of unlabelled text, followed by fine-tuning for specific NLP tasks. During pre-training, MLM is used to mask random tokens and predict them. At the same time, NSP is used to predict if two given sentences are sequential. This bidirectional pre-training allows BERT to understand the context from both directions, providing richer understanding compared to unidirectional language models.

2.1.2 DistilBERT

DistilBERT is a lightweight version of BERT that reduces model size while retaining most of BERT's language understanding capabilities and improving inference speed. This model leverages a technique known as Knowledge Distillation to effectively learn during the pre-training stage, using significantly fewer computational resources compared to the original BERT model, while still delivering excellent performance.

$$L_{ce} = \Sigma_i t_i * \log(s_i) \qquad (2)$$

Knowledge Distillation is a technique that transfers the knowledge of a larger model (teacher model) to a smaller model (student model). In Equation (2), t_i represents the probability from the teacher model, and s_i represents the probability from the student model. The loss function helps the student model to accurately replicate the output

distribution of the teacher model. During this process, the student model learns to mimic the soft probability distribution produced by the teacher model. The prediction probability distribution from the teacher model can be adjusted using a parameter called Softmax temperature. The Softmax temperature is used to smooth out the model's output, where a higher value produces a smoother distribution (with probabilities more evenly spread), and a lower value makes it sharper (with a higher probability for specific classes).

DistilBERT retains the core BERT architecture but reduces its size by removing token type embeddings and the pooler and by halving the number of layers, thus reducing the model size by 40%. This reduced number of layers greatly enhances the model's computational efficiency. Additionally, DistilBERT is initialized using intermediate layers from the teacher model, enabling it to effectively leverage the context learned by the teacher model.

DistilBERT's training employs a combination of three loss functions: Masked Language Modeling Loss (L_{MLM}), Distillation Loss (L_{CE}), and Cosine Embedding Loss (L_{COS}). These three loss functions help the model to effectively imitate not only the linguistic characteristics of the text but also the teacher model's performance.

Masked Language Modeling Loss (L_{MLM}) is a technique originally used in BERT pre-training, where parts of sentences are masked, and the model is trained to predict the masked words. For instance, in the sentence, "He ate [MASK] this morning," the model learns to correctly predict the masked word as 'apple,' 'banana,' or other plausible options.

Distillation Loss(L_{CE}) is a loss function that helps the student model to mimic the output distribution from the teacher model. The student model aims to predict each class's Softmax probability as closely as possible to the teacher model. For example, if the teacher model predicts a "positive" sentiment with a probability of 0.7, and the student model predicts the same sentiment with a probability of 0.5, the training process will minimize this difference.

Cosine Embedding Loss (L_{COS}) ensures that the embedding vectors of the student and teacher models point in similar directions. The goal is to keep the embedding spaces of the two models close to each other by minimizing the angle between the vectors. For instance, if the embedding vector of the teacher model points in a particular direction, the student's embedding vector is adjusted to point as closely as possible in that direction.

Through this combination of loss functions, DistilBERT can effectively reduce model size and computational cost without significantly sacrificing the teacher model's performance.

2.2 Previous Studies

2.2.1 TTQ

TTQ (Trained Ternary Quantization) reduces model size and enhances computational efficiency by representing neural network weights using only three values (-W, 0, +W). Each layer employs two scaling coefficients to quantize weights into ternary values, with these coefficients being optimized during the training process. For each weight w, the quantized weight w_t takes one of three possible values:

$$w_t \begin{cases} +W_p & \text{if } w > \Delta \\ 0 & \text{if } |w| \leq \Delta \\ -W_n & \text{if } w < -\Delta \end{cases} \quad (3)$$

In Equation (3), W_p and W_n denote scaling coefficients for positive and negative weights, respectively, while Δ represents the threshold. For example, if a neural network layer's weight vector is given as [-1.5, 0.2, 0, -0.3, 1.2] and the threshold Δ is set to 0.5, the quantized weights become [$-W_n$, 0, 0, 0, W_p]. If W_p = 1 and W_n = 1, the final weights are quantized to [-1, 0, 0, 0, 1]. During training, the model retains full precision weights w while updating W_p and W_n by computing gradients.

2.2.2 FGQ

FGQ (Fine-Grained Quantization) is a method that converts traditional models into ternary format. This technique transforms the model weights into ternary values while minimizing classification accuracy loss, achieving top performance on the ImageNet dataset without additional training. FGQ reduces computational complexity by finding an optimal scaling factor α using the Frobenius norm to minimize the error between the ternarized weights and the original weights.

$$\|A\|_F = \sqrt{\Sigma_{i,j} a_{i,j}^2} \quad (4)$$

$$\alpha^*, \beta^* = \text{argmin}_{\alpha \geq 0, \beta 0} E(\alpha, \beta),\ E(\alpha, \beta) = \|W - \alpha \widehat{W}\|_F^2 \quad (5)$$

In Equation (4), $\|\cdot\|_F$ represents the Frobenius norm, calculated as the square root of the sum of the squares of a matrix's elements. This is equivalent to the Euclidean norm when matrix A's elements are flattened into a vector.

In Equation (5), α(scaling factor) is used during the ternarization of each weight in W. This factor scales the original weights to minimize the difference from the ternary weights $\widehat{W}$, where $\widehat{W}$ is the ternarized form of the original weights W, converting each element to -1, 0, or +1. The conversion criteria are determined primarily by a threshold β. For example, if the weight vector is W =[0.5, -1.5, 0.3, 2.1] and the threshold β=1, ternarization proceeds as follows: |0.5|<β → 0, |-1.5|>β → -1, |0.3|<β → 0, |2.1|>β → +1. Thus, $\widehat{W}$ = [0, -1, 0, 1]. Next, the scaling factor α is calculated as a parameter determined automatically during model optimization. Finally, the Frobenius norm is calculated and to simplify calculations and facilitate optimization (e.g., for easier differentiation), the squared Frobenius norm (Equation 6) is often used.

$$\|W - \alpha \widehat{W}\|_F^2 = \|(0.5, -1.5, 0.3, 2.1) - \alpha(0, -1, 0, 1)\|^2 \quad (6)$$

This value depends on α and methods such as differentiation are employed to find the optimal α. This process allows ternarized weights to closely resemble the original weights, minimizing overall performance degradation in the neural network.

2.3 Differences from Previous Studies

TTQ is a technique that reduces the model size and improves computational efficiency by representing neural network weights with only three values: -W, 0, and +W. It quantizes the weights of each layer into ternary values and optimizes them during training. However, TTQ limits the dynamic range of weights and lacks flexibility in specific weight values, which can lead to performance degradation. This restriction makes it challenging to apply TTQ to various neural network structures or language models.

FGQ is a method that minimizes performance loss by converting a trained model into ternary form. It optimizes a scaling factor using the Frobenius norm to minimize the error between original weights and ternarized weights. However, FGQ has limitations, as it requires a complex optimization process and may not be suitable for real-time applications. Additionally, FGQ processes the weights of the entire model in the same manner, which may fail to fully capture the unique characteristics of specific layers.

Unlike these conventional quantization methods, TOQ applies ternary quantization only to the output data, reducing computational complexity while enabling the model to focus on essential information. This approach simplifies the output data into three states(-1, 0, +1) to enhance computational efficiency and create a structure suitable for real-time processing. Consequently, TOQ addresses the issue of limited dynamic range in previous methods and can be flexibly applied to various neural network architectures. Furthermore, TOQ efficiently processes information in the final layer, reducing computational load and enhancing processing speed. By focusing on the output layer rather than the internal model layers, TOQ maximizes performance gains and resource savings in real-world applications. This approach suggests that TOQ can overcome the limitations of previous studies and make a significant contribution to improving the training and efficiency of large-scale language models, especially in resource-constrained environments.

3. Principle of TOQ Implementation

3.1 Basic Principle of TOQ

TOQ simplifies the continuous output values of the model into three discrete states (-1, 0, +1). The quantization is performed for each neural network output z_i based on the following conditions. Equation (7) presents an example of Ternary Quantization when the threshold is set to 0.5.

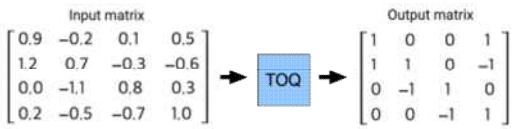
$$Q(z_i) = \begin{cases} -1 & \text{if } z_i < -0.5 \\ 0 & \text{if } -0.5 \leq z_i \leq 0.5 \\ +1 & \text{if } z_i > 0.5 \end{cases} \quad (7)$$

In this context, $Q(z_i)$ represents the quantized output value. This simplified output value guides the

model to focus more attention on certain parts during the weight update process. For instance, -1 and 1 indicate that the model should assign a larger or smaller weight to the corresponding output, prompting the model to concentrate on important information, discard unnecessary information, and diminish the impact of negative information. Unlike traditional continuous weight adjustments, this discrete quantization approach enhances computational efficiency, helping to maintain model performance, particularly in resource-constrained environments.

3.2 Weight Adjustment Mechanism

In a model where TOQ is applied, the quantized output values are used to adjust the weights during the training process. A value of 1 signifies that the feature is considered more important in the learning process, leading to positive reinforcement by increasing its weight. A value of 0 suggests that the feature does not significantly contribute to the weight update and can be neutral or disregarded. A value of -1 indicates that the feature may have a negative impact, prompting the model to reduce its influence. In the context of classification tasks, this ternary output affects how the model adjusts its internal weights, which in turn influences the way the model makes predictions. This mechanism allows the model to learn more efficiently and to focus on critical information, enhancing its performance and prioritizing essential features.



(Image 2) Ternary Output Quantization

4. Experiment and Results

4.1 Study

4.1.1 Research Environment and Data

This study was conducted on a MacBook Pro equipped with Apple's M1 Max chip (10-core CPU, 24-core GPU, 16-core Neural Engine, and 32GB unified memory). The software used was Jupyter Notebook from the Anaconda distribution.

The research data used was the 2023 newspaper corpus provided by the National Institute of the Korean Language. This dataset encompasses various contemporary Korean usage examples and reflects real-world language usage patterns. For training data, the seventh file in the "Newspaper Regional Comprehensive Media Article Raw Corpus" (NLRW2300000007.json) was used, and for validation data, the second file (NLRW2300000002.json) was utilized.

4.1.2 Text Preprocessing and Model

For preprocessing Korean data, we utilized the Okt object from the Konlpy library. Konlpy is a natural language processing library specifically designed for Korean, which is particularly useful for complex morphological analysis and part-of-speech tagging in Korean. During the preprocessing stage, special characters were removed from the Korean text, and only necessary nouns were extracted for model training.

BERT, developed by Google, is a model designed to process input data bidirectionally, enabling it to understand context more effectively. This model shows high performance, especially when handling large-scale language data, and is widely used in various natural language processing tasks. DistilBERT is a lightweight version of BERT that reduces the model size while improving training speed, maintaining performance close to the original model.

In this study, we developed a custom model based on the well-established BERT and DistilBERT models, incorporating the TOQ module. The TOQ module is applied to the output layer of the model, maximizing training efficiency by quantizing the output data.

4.1.3 Experimental Design and Evaluation Method

The main task used in this study is MLM. MLM randomly masks certain words in a sentence and evaluates the model's language comprehension by asking it to predict the missing words. MLM plays a critical role in enhancing the language model's prediction capability and assessing the appropriateness of words within a given context.

In this study, the effectiveness of TOQ was quantitatively evaluated by comparing models with and without the TOQ module (bert-base-uncased and

distilbert) on the MLM task. The evaluation metrics include Training Loss, Validation Loss, and Training Speed. Training Loss indicates how well the model is learning from the training data. By comparing the Training Loss between the TOQ-applied model and the baseline model, we can assess the effectiveness of TOQ. Validation Loss measures how well the model generalizes to the validation data, helping us analyze the impact of TOQ's quantization on the model's generalization capability. Lastly, Training Speed is assessed by measuring the time taken for model training, which shows the impact of TOQ on training efficiency. This metric is particularly significant in environments with limited computing resources.

4.2 Experimental Results

The experimental results of this study demonstrate that TOQ improves the learning and generalization capabilities of both BERT and DistilBERT models. For instance, the DistilBERT model with TOQ applied shows improved Training Loss and Validation Loss compared to the baseline model, while significantly reducing training time. This provides considerable advantages for training and deploying language models, particularly in environments with limited computational resources.

(Table 1) Results of BERT and DistilBERT Experiments

Model	Epochs	Batch size	Gradient accumulation steps	Learning rate	Training Loss	Validation Loss	Train runtime(sec)
BERT	0.5	16	4	3e-3(0.003)	3.5572	3.5414	11,865
BERT+TOQ	0.5	16	4	3e-3(0.003)	3.6289	3.7412	8,310
BERT	3	16	4	3e-3(0.003)	3.5359	3.5379	81,021
BERT+TOQ	3	16	4	3e-3(0.003)	3.3941	3.5932	56,156
DistilBERT	3	16	4	3e-3(0.003)	0.1046	0.1123	57,812
DistilBERT +TOQ	3	16	4	3e-3(0.003)	0.0998	0.1039	24,326
DistilBERT	3	8	4	3e-3(0.003)	0.1045	0.1123	52,599
DistilBERT +TOQ	3	8	4	3e-3(0.003)	0.0987	0.1039	42,588

In the case of 0.5 epochs, despite the short training duration, TOQ demonstrated some effectiveness in reducing training time. During 0.5 epochs, the average training loss of the BERT model was 3.5572, which increased to 3.6289 after applying TOQ, suggesting that the model experienced slight challenges during the early training phase. However, after completing 3 epochs of training, the training loss of the TOQ-applied BERT model improved to 3.3941 compared to the original model, and the validation loss was also comparable to that of the original model. This indicates that while TOQ may initially pose difficulties for the model during early training, it enhances the model's learning ability over the long term. Ultimately, TOQ proves its potential as an efficient language model training method by enhancing model performance and reducing training time. One of the primary advantages of TOQ is its ability to facilitate fast training and efficient deployment, especially in resource-constrained environments.

Through empirical validation, this study has shown that TOQ optimizes the balance between computational efficiency and model performance, enabling practical applications of large-scale language models. This finding is particularly significant in resource-limited environments, as TOQ not only improves model learning and generalization capabilities but also reduces model size and computational demands, greatly enhancing usability in real-world scenarios. In conclusion, this study demonstrates that the TOQ method is an effective way to reduce training time and improve model performance in the training and validation processes of language processing models. It is anticipated to make a valuable contribution to the development and application of large-scale language models in the future.

5. Conclusion and Future Research Directions

In this study, we applied the Ternary Output Quantization (TOQ) module to BERT and DistilBERT models, demonstrating its potential to reduce training time and enhance model performance. Experimental results indicated that TOQ can effectively reduce loss rates while improving training speed, particularly showcasing its potential to enhance model efficiency in environments with limited computational resources. These findings may be especially valuable in fields such as cloud computing and real-time language processing, where performance improvement is crucial.

However, there are two limitations to this study.

First, the current TOQ module is primarily limited to specific types of language models, necessitating further evaluation of its applicability and effectiveness across diverse model types. Second, there is a lack of research on how TOQ impacts model interpretability, which poses challenges in fully understanding the model's decision-making processes. Future research should focus on validating the generalizability of the TOQ module by applying it to various types of neural network models. Studies will explore the effects of TOQ on generative models like GPT, multimodal models such as CLIP, and image processing models like CNNs. This will enable a more comprehensive evaluation of how TOQ can improve performance in a range of real-world applications.

In conclusion, this study demonstrates that the TOQ technique is an effective method for reducing training time and enhancing model performance in language processing models, with promising implications for the future development and application of large-scale language models.

Reference

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. (2017), Attention is All You Need, *Advances in Neural Information Processing Systems, 30.*

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. (2020), Language models are few-shot learners, *arXiv preprint arXiv:2005.14165.*

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. and Amodei, D. (2020), Scaling laws for neural language models, *arXiv preprint arXiv:2001.08361.*

Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanty, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., Phanishayee, A. and Zaharia, M. (2021), Efficient large-scale language model training on GPU clusters, *arXiv preprint arXiv:2104.04473.*

Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W. (2021), LoRA: Low-Rank Adaptation of Large Language Models, *arXiv preprint arXiv:2106.09685.*

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S. and Kiela, D., (2020), Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, *arXiv preprint arXiv:2005.11401.*

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, *arXiv preprint arXiv:1810.04805*

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019), DistilBERT: A distilled version of BERT: smaller, faster, cheaper and lighter, *arXiv preprint arXiv:1910.01108*

Mellempudi, N., Kundu, A., Mudigere, D., Das, D., Kaul, B. and Dubey, P. (2017), “Ternary Neural Networks with Fine-Grained Quantization”, *arXiv preprint arXiv:1705.01462v3*

Zhu, C., Han, S., Mao, H. and Dally, W. J. (2016), “TRAINED TERNARY QUANTIZATION”, *arXiv preprint arXiv:1612.01064v3.*

Joshua K. Cage, 임선집, 채호창(2023), “101가지 문제로 배우는 딥러닝 허깅페이스 트랜스포머 with 파이토치,” 32-90.

◐ ABOUT THE AUTHOR ◑

Do-hoon Lee

Enrolled in the Master's in AI·Big Data program at the Graduate School of AI, Seoul School of Integrated Sciences & Technologies (aSSIST)

Currently working at the Youth Counseling and Welfare Center, Gyeongsangnam-do Youth Support Foundation

Areas of interest : AI, Data Analysis etc.

E-mail : ehgnsslekd@naver.com