Fine-Tune the 1-Bit Vision Transformer on CIFAR-10

[1,*]*Yu-Lun Liu* (劉宇倫), [1]*Chiou-Shann Fuh* (傅楸善),

[1]Graduate Institute of Biomedical Electronics and Bioinformatics,
National Taiwan University, Taipei, Taiwan,

[*]E-mail: r12945054@ntu.edu.tw fuh@csie.ntu.edu.tw

ABSTRACT

This paper presents an evaluation of a 1-bit quantized [1] Vision Transformer (ViT), fine-tuned on the Canadian Institute For Advanced Research (CIFAR-10) dataset, a standard benchmark in image classification consisting of 60,000 images of 32x32 pixels color images in 10 different classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Using a pre-trained ViT and BitLinear for quantization, we assess the trade-off between model size, computational efficiency, and accuracy.

Our findings indicate that the 1-bit ViT, while having a lower accuracy of 40.5%, significantly reduces computational complexity and achieves an exceptionally low inference time of 0.010 ms. This makes it ideal for applications demanding high speed and limited computational resources.

Keywords: *1-bit quantization, Vision Transformer (ViT), CIFAR-10, image classification, model compression, BitLinear, computational efficiency, classification accuracy.*

1. INTRODUCTION

The Vision Transformer (ViT) [2] has emerged as a powerful architecture in the field of computer vision, demonstrating remarkable performance on various image recognition tasks. The model's ability to capture global dependencies through the self-attention mechanism [3] has been a key factor in its success. However, the deployment of such large-scale models in resource-constrained environments poses significant challenges due to their high computational and memory requirements.

Quantization, specifically 1-bit quantization, offers a promising solution by simplifying the model's arithmetic operations and shrinking its size. BitLinear, a custom quantization technique, is employed to convert the model's weights and activations into binary format. This paper outlines the fine-tuning process of the pre-trained ViT using BitLinear, adapting it to the CIFAR-10 dataset, and evaluates the performance implications of this quantization.

2. METHODOLOGY

This section delves into the specifics of the BitNet architecture, the quantization process, and the fine-tuning procedure. It describes the steps taken to adapt the pre-trained ViT model to the CIFAR-10 dataset using the BitLinear layer and details the training regimen, including data preprocessing, optimization strategy, and evaluation metrics.

2.1 Vision transformer architecture

The Vision Transformer (ViT) model is an innovative approach to image classification that leverages a transformer architecture, originally popularized by Natural Language Processing (NLP) tasks, to work on images. The model was introduced in the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" by Alexey Dosovitskiy et al. [2]

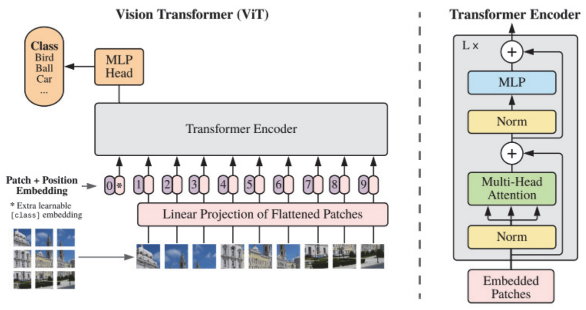

Fig. 1. Model overview. We divide an image into fixed-size patches, then linearly embed each patch. After that, we add positional embeddings to these vectors and input the resulting sequence into a standard Transformer encoder. For classification tasks, we follow the typical method of appending an additional learnable "classification token" to the sequence.

Here's a brief overview of its architecture:

Input Representation: The model takes images as input, which are divided into fixed-size patches. Here, we use 224x224x3 pixels as the image size. Each patch is flattened into a single vector, and these vectors are then linearly transformed into embeddings (like word embeddings in NLP).

Positional Encoding: Since transformers lack an inherent sense of order, positional encodings are added to the input embeddings to retain information about the relative positions of the patches within the image.

Transformer Encoder: The model uses a standard transformer encoder architecture, consisting of layers of multi-head self-attention and feedforward neural networks. These layers process the sequence of input embeddings and learn complex patterns in the data.

Classification Token: A special classification token (or [CLS] token) is prepended to the input sequence. After processing through the transformer layers, the representation of this token is used as the final output of the model for classification.

Output: The representation of the [CLS] token is passed through a final linear layer to produce the class probabilities for the input image.

The core of the self-attention mechanism lies in calculating the relationship weights between each patch and all other patches, indicating the extent to which other patches should be considered in generating the representation for each patch. This mechanism enables the model to dynamically focus on the most significant parts of an image, thereby enhancing its ability to understand images.

2.2 Quantization with BitLinear

In terms of quantization performance, quantization-aware training has minimal impact on model performance and allows for continued training or fine-tuning of the model. The implementation of BitNet is straightforward, simply requiring the replacement of linear projections in the Transformer by substituting nn.Linear in PyTorch with BitLinear. This quantization method can significantly reduce the model's computational complexity and memory usage by converting floating-point operations to simple binary operations.

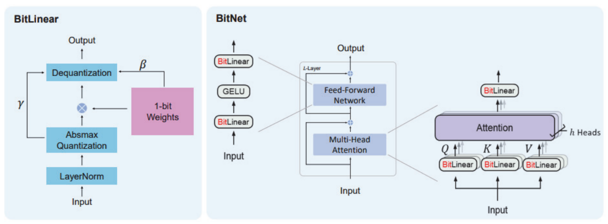

Fig. 2. On the left is the computation flow of BitLinear, on the right is the basic architecture of BitNet. [1]

First, we convert the weights to either +1 or -1 using the signum function. We then adjust the weights so that their average is zero before converting them to binary. This helps make better use of the available numerical range. After converting the weights to binary, we apply a scaling factor, β, to minimize the difference between the original real-valued weights and the new binary weights. In our work, we select β as 8.

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

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

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

Fig. 3. The process of determining binary values from the weight matrix W. The weights are divided by the shape of the matrix to obtain α, then the difference from the original weights is calculated. The binary values are determined based on the sign of the new weights.

We convert the activations to b-bit precision by absmax quantization. This scales the activations into the range [$-Q_b$, Q_b] (where $Q_b = 2^{b-1}$) by multiplying them using Q_b and dividing by the maximum absolute value of the input matrix. Here, ϵ is a small floating-point number that helps prevent overflow during clipping.

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

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

Fig. 4. The quantization of activation values where γ represents the absolute maximum value of the input matrix, while b denotes the set precision level; Q_b is calculated as 2^{b-1}. The input is scaled by Q_b and then divided by γ, with the range controlled by a clipping function.

For the activations before applying non-linear functions, we adjust them so that they fall within the range [0, Q_b]. We do this by subtracting the minimum value from the inputs to make all values non-negative.

$$\widetilde{x} = \text{Quant}(x) = \text{Clip}((x - \eta) \times \frac{Q_b}{\gamma}, \epsilon, Q_b - \epsilon), \quad \eta = \min_{ij} x_{ij}.$$

Fig. 5. The quantization method outlined in Fig. 3. by first subtracting the minimum value of the matrix η.

$$y = \widetilde{W}\widetilde{x}$$

$$\begin{aligned}\mathrm{Var}(y) &= n\mathrm{Var}(\widetilde{w}\widetilde{x}) \\ &= nE[\widetilde{w}^2]E[\widetilde{x}^2] \\ &= n\beta^2 E[\widetilde{x}^2] \approx E[\widetilde{x}^2]\end{aligned}$$

Fig. 6. The matrix multiplication and the variance of the output y. Here we assume that W and x are mutually independent, follow the same distribution, and are independent of each other; n is the number of elements; w_e are the elements of W after scaling; x_e are the quantized activations, and β is the scaling factor used during the weight binarization.

For full-precision computation, the variance of the output $\mathrm{Var}(y)$ is typically around 1 when using standard initialization methods like Kaiming or Xavier initialization, which helps maintain training stability. To preserve this variance after quantization, we introduce a LayerNorm [4] function before activation quantization. This ensures that the variance of the output y remains approximately 1, matching the full-precision counterpart's variance. In the context of Transformers, this implementation is known as SubLN [5].

$$y = \widetilde{W}\widetilde{x} = \widetilde{W}\,\mathrm{Quant}(\mathrm{LN}(x)) \times \frac{\beta\gamma}{Q_b}$$

$$\mathrm{LN}(x) = \frac{x - E(x)}{\sqrt{\mathrm{Var}(x) + \epsilon}}, \quad \beta = \frac{1}{nm}||W||_1$$

Fig. 7. The BitLinear Equation.

2.3 Fine-Tuning on CIFAR-10

Fine-tuning a pre-trained Vision Transformer (ViT) model involves adapting the model to a new task while retaining the learned features from the pre-training phase. We fine-tuned a pre-trained ViT model on the CIFAR-10 dataset, including the preparation steps, the training strategy, and the final training phase.

2.4 Steps

1. Load a pre-trained ViT model and feature extractor from the Hugging Face transformers library [6].
2. Define a custom 1-bit fully connected layer (BitLinear) to replace the original classifier in the ViT model.
3. Load the CIFAR-10 dataset [7] and resize the images to 224x224 pixels to match the input requirements of the ViT model.
4. Split the dataset into training, validation, and test sets.
5. Define the image transformation process, including resizing, converting to tensors, and normalization.
6. Train the model using the SGD optimizer and cross-entropy loss function. During training, print the loss value every 200 batches and evaluate the model's performance on the validation set at the end of each epoch.

3. RESULTS

In this paper, we examine the performance of various deep learning models for image classification, highlighting the trade-offs between accuracy, computational complexity, and inference speed. Our analysis includes four models: the 1-bit Vision Transformer (1-bit ViT), Vision Transformer (ViT), Resnet 18 [8], and VGG16 [9]. Each model presents unique advantages depending on the application requirements, such as accuracy, model size, computational complexity, and speed.

ResNet, or Residual Network, introduces the concept of residual learning to facilitate the training of deeper neural networks. ResNet 18, the 18-layer variant, is renowned for its efficiency and performance across a variety of tasks. It utilizes skip connections or shortcuts to jump over some layers, preventing the vanishing gradient problem and allowing the network to train faster and more effectively. This architectural choice results in a notable reduction in model complexity and inference time compared to deeper models, making it a popular choice for real-time applications.

VGG16 is a variant of the Visual Geometry Group network that consists of 16 layers. It is characterized by its simplicity, using only 3x3 convolutional layers stacked on top of each other in increasing depth. Despite its relatively larger size and computational requirements compared to more modern architectures, VGG16 has proven to be very effective for image recognition tasks, offering competitive accuracy levels. Its extensive use of convolutional layers, however, translates into significant computational demand and memory requirements.

3.1 The evaluation score

The evaluation metrics are defined as follows:

Model size: Comparison of the number of model parameters and required storage space.

Inference time: Measurement of the time taken for the model to make a prediction on a single image.

Computational complexity: Evaluation of the number of FLoating-point OPerations (FLOPs) required to perform one forward pass of the model.

Table 1. Performance comparison of different neural network architectures.

	1-bit ViT	ViT	Resnet 18	VGG16
Accuracy↑	0.405	**0.986**	0.948	0.936
Model size (MB)↓	327.33	327.33	**42.65**	512.32
Inference time(ms)↓	**0.010**	0.297	0.054	0.032
Computational complexity ↓	17.59 GMac	17.59 GMac	**1.82 GMac**	15.52 GMac
Number of parameters ↓	85.81 M	85.81 M	**11.18 M**	134.3 M

The Vision Transformer (ViT) stands out with an exemplary accuracy rate of 98.6%, showcasing the strengths of transformer architectures in capturing intricate image features for classification. This performance underscores the potential of transformer-based models in handling complex image analysis tasks.

However, when computational efficiency is paramount, the 1-bit Vision Transformer offers an appealing alternative. Despite its lower accuracy of 40.5%, the 1-bit ViT dramatically reduces computational complexity while achieving an exceptionally low inference time of 0.010 ms. This efficiency makes it suitable for applications requiring high speed with constrained computational resources.

Resnet 18 strikes an optimal balance between model performance and efficiency. With an accuracy of 94.8%, it maintains a relatively small model size of 42.65 MB and a moderate computational complexity of 1.82 GMac. This balance positions Resnet 18 as a practical choice for a wide range of applications where both accuracy and computational efficiency are considered.

In contrast, VGG16, despite its competitive accuracy of 93.6%, demands significant computational resources, including the largest model size of 512.32 MB and a computational complexity of 15.52 GMac. While VGG16's powerful feature extraction capability makes it a strong candidate for accuracy-focused tasks, its resource requirements may limit its usability in resource-constrained environments.

In conclusion, selecting an appropriate model for image classification tasks depends on specific application needs. While ViT offers superior accuracy, its computational demands may not suit all scenarios. The 1-bit ViT provides an efficient solution for speed-critical applications, albeit at lower accuracy. Resnet 18 represents a middle ground, offering a good mix of accuracy and efficiency, making it suitable for a variety of applications. Meanwhile, VGG16, despite its high computational cost, remains a viable option for tasks where accuracy is the primary concern. This analysis underscores the importance of considering multiple factors, including accuracy, inference speed, and computational resources, when choosing a deep learning model for image classification.

3.2 1-bit vision transformer classification examples

To provide a clearer picture of the practical performance of the 1-bit Vision Transformer, we have curated a selection of classification instances. These instances are divided into two categories: ten examples where the model accurately classified the images, demonstrating its effectiveness in certain scenarios; and ten examples where the model incorrectly classified the images, highlighting potential areas for improvement or the limits of its binary-weight architecture.

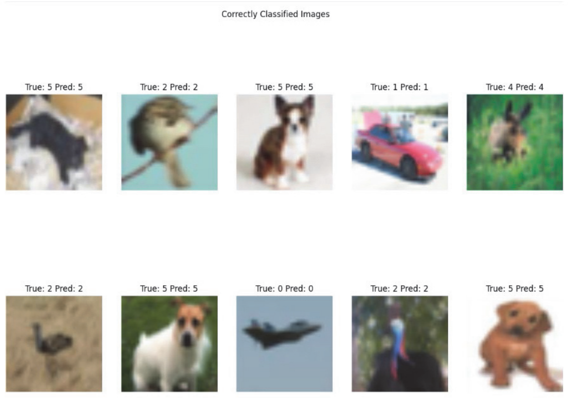

Fig 8. Here we illustrate examples of accurate classifications made by the 1-bit Vision Transformer, showcasing its proficiency in correctly identifying images across a diverse set of categories. These examples span the following labels: 0. Airplane, 1. Automobile, 2. Bird, 3. Cat, 4. Deer, 5. Dog, 6. Frog, 7. Horse, 8. Ship, 9. Truck. This figure serves as a testament to the model's capabilities in specific instances, despite the challenges it faces in precision detailed in other aspects of its evaluation.

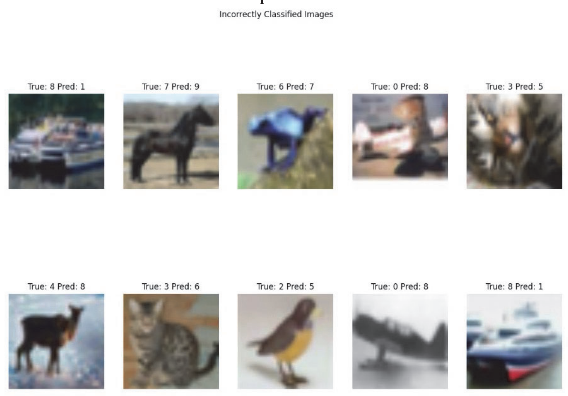

Fig 9. Incorrect classification examples by 1-bit vision transformer.

These examples serve as a valuable tool for understanding the real-world applicability of the 1-bit Vision Transformer, revealing patterns in the types of images that are either handled well by the model or pose challenges

3.3 Confusion matrices across models

Further, to offer a more comprehensive analysis of each model's classification performance, we examine the confusion matrices for the 1-bit Vision Transformer, Vision Transformer(ViT), Resnet 18, and VGG16. Confusion matrices provide insight into not only the overall accuracy of each model but also detail how each class is being predicted, identifying potential biases or weaknesses towards certain categories.

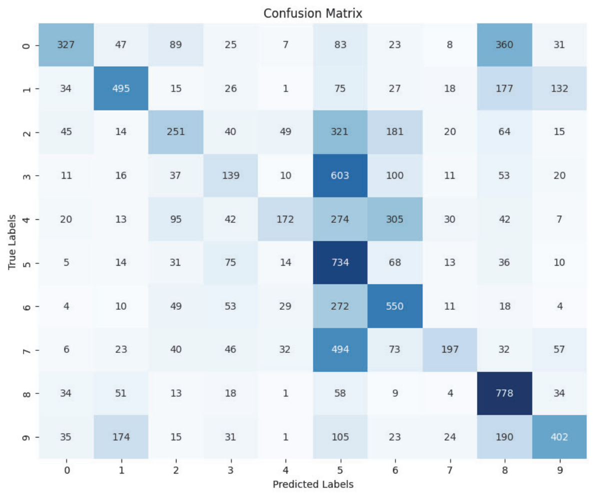

Fig. 10. Confusion matrix for 1-bit Vision Transformer (ViT)

The confusion matrix for the 1-bit Vision Transformer (ViT), as exhibited in Figure 10, underscores the model's challenge in achieving precise classification across an array of categories. This is particularly manifested through the substantial number of misclassifications highlighted by the off-diagonal elements within the matrix. The 1-bit ViT model, designed to operate with reduced precision to potentially increase computational efficiency, unfortunately faces significant limitations in its ability to accurately capture and distinguish detailed features inherent to various classes. This limitation becomes notably evident in its handling of classes 3(cat) and 5(dog).

In addition to the pronounced misclassifications observed in classes 3 and 5, the matrix further reveals a tendency of the 1-bit ViT model to confuse classes 0 and 8, synonymous with aircraft and ships respectively. This confusion between such distinct categories reflects a deeper issue within the model's feature detection and classification mechanisms, suggesting a particular struggle in discerning between objects that share certain visual characteristics at a high level, such as shape and context within their respective environments.

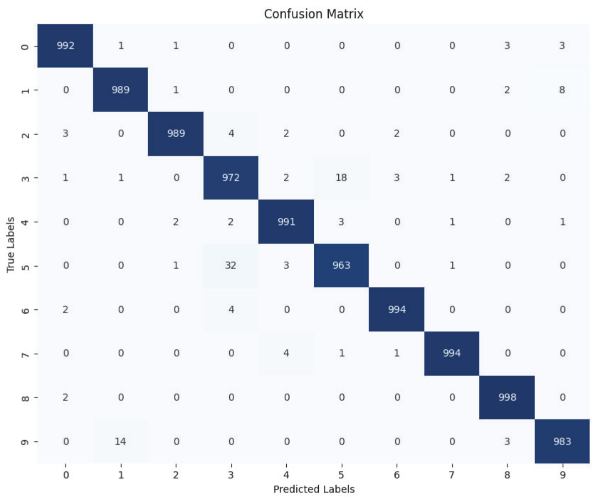

Fig. 11. Confusion matrix for Vision Transformer (ViT)

The standard ViT model[Fig. 11] demonstrates a high degree of accuracy, with most predictions concentrated along the diagonal, indicating correct classifications. There are minimal misclassifications, with a few instances of confusion, such as class 8(bird) being misclassified as class 2(ship), which are relatively rare.

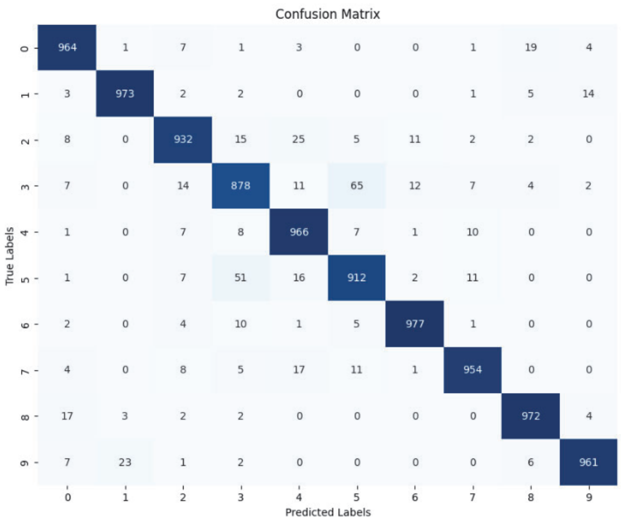

Fig. 12. Confusion matrix for Resnet 18

ResNet 18 also shows high classification accuracy with most values along the diagonal, though not as high as ViT. In this confusion matrix, we can identify that classification errors occur in the 3rd and 5th categories, which are cats and dogs, respectively.

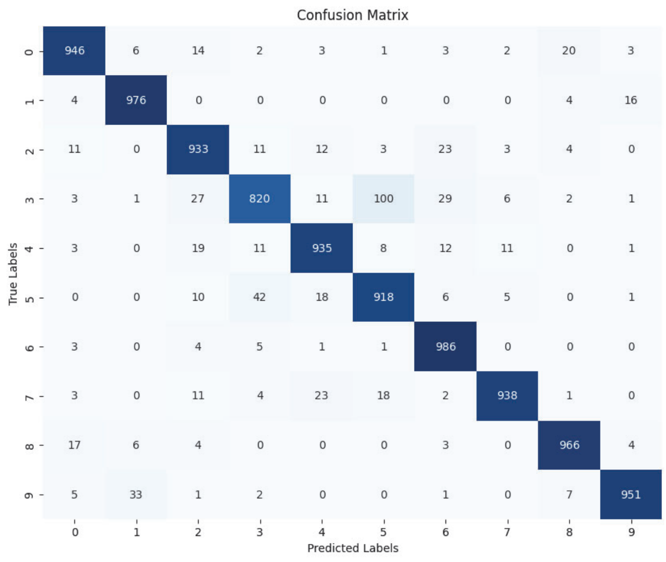

Fig. 13. Confusion matrix for VGG16

The VGG16 model demonstrates slightly inferior performance in terms of accuracy relative to the Vision Transformer (ViT) and ResNet 18. This dip in precision is notably pronounced for categories 3 and 5, mirroring a pattern of classification errors like those observed in the previous models.

In summary, the examination of confusion matrices for the 1-bit Vision Transformer, Vision Transformer (ViT), ResNet 18, and VGG16 provides a detailed comparative analysis of the classification performance of each model. This analysis reveals that:

1. The 1-bit Vision Transformer struggles with significant misclassifications across various classes, with pronounced errors in classes 3 and 5, indicating challenges in feature detection with reduced precision.
2. The standard Vision Transformer displays a robust classification accuracy, with most predictions correctly aligned along the diagonal, showcasing minimal misclassifications and only isolated instances of confusion between specific classes.
3. ResNet 18 demonstrates high accuracy, albeit slightly lower than the ViT model. Notable misclassifications were observed in classes associated with cats and dogs, showing a specific area where improvement is needed.
4. The VGG16 model experiences a slight reduction in accuracy compared to the Vision Transformer and ResNet 18, with classification errors in classes 3 and 5 mirroring errors seen in other models.

By meticulously analyzing these confusion matrices, it becomes apparent how each model performs across different classes, highlighting potential biases or weaknesses. This comparative study not only helps in understanding the specific misclassification patterns but also underscores areas requiring attention for optimization and refinement. Consequently, this investigation enriches our strategic approach to model selection, tailoring our choices to meet the nuanced needs of diverse applications in image classification. Through this comprehensive evaluation, we gain valuable insights into the strengths and limitations of each model, paving the way for targeted improvements and more informed decision-making in the deployment of deep learning models for image classification tasks.

4. Discussion

4.1 Quantization and model Size: a detailed examination

Upon revisiting the entire code block, the reasons why the model size might not change are speculated as follows:

Quantization Implementation Method:
In the BitLinear class, even though the weights and inputs are subjected to quantization operations (converting weights to 1 or -1), these operations are conducted dynamically during the forward propagation process of the model. This implies that while quantized values are utilized during the operational process, the weights of the model itself are still stored in full precision (as 32-bit floating-point numbers) in memory. Consequently, this method of quantization does not directly reduce the size of the model when stored.

Model Storage Format:
The size of a model depends not only on the data type of the weights but also on the format used for model storage. If the model is stored in a format that does not support quantized weights (for example, the standard PyTorch model storage format), then the size of the stored model will not be reduced, even though the weights are quantized during operations.

Data Type after Quantization:
Even if the weights are quantized during the forward process, if the quantized data is not converted to a data type that occupies less space (for example, from float32 to int8 or uint8), there will not be a significant reduction in the size of the model.

This analysis suggests that for a substantial reduction in model size through quantization, it is imperative not only to implement quantization operations during model computation but also to consider the storage format and ensure that quantized weights are stored in an optimized data type that requires less space.

4.2 The impact of training data volume on model accuracy

In the research papers currently available to us, teams using 1-bit or 1.58-bit BitLinear[10] structures have applied them to language models, with the input data volume starting at a minimum of 100M. Therefore, we speculate that the insufficient volume of our training data might be the reason why, after applying the BitLinear structure, we lost too much detail and were unable to achieve the same level of accuracy as the original model.

This speculation is based on the premise that a large volume of training data can provide richer information, helping the model to learn more features and details, thereby improving the model's prediction accuracy. Conversely, if the volume of training data is insufficient, even with an efficient model structure, the expected performance might not be achieved due to a lack of adequate learning material.

Given the significant impact of data volume on model performance, we may need to further collect and augment our training data, or try using model architectures that are more suited for smaller datasets, in hopes of improving the accuracy performance of models adopting the BitLinear structure.

5. Conclusion

In the pursuit of advancing neural network models, particularly in the domain of computer vision and language processing, the exploration of quantization techniques, such as 1-bit or 1.58bit networks, has emerged as a pivotal area of research.

The introduction of quantization methods, particularly the transformation of traditional models to their 1-bit counterparts through structures like BitLinear, has shown promise in reducing model size and computational complexity significantly. These advancements not only make models more accessible for deployment on resource-constrained environments, such as edge devices, but also open new vistas for sustainable AI solutions.

However, this transformation comes with its caveats. The precision loss inherent in quantization unavoidably leads to a potential drop in model accuracy, a challenge that has been observed across various implementations, underscoring the need for a balanced approach to quantization that carefully considers the trade-off between efficiency and accuracy.

Analysis has revealed that one of the potential reasons behind the accuracy dip in quantized models, compared to their full-precision counterparts, could be attributed to the volume of training data. Most successful applications of bit-quantized networks have been on language models with substantial training data volumes (starting from 100M), implying a direct correlation between data volume and the model's ability to retain accuracy post-quantization.

This suggests that while quantization optimizes models for speed and size, the sufficiency of training data becomes even more critical to ensure that the reduced precision does not severely hinder model performance. Therefore, for applications with limited data, it's important to explore techniques that can augment the data or optimize the model structure specifically for low-resource settings.

Our work towards optimized neural networks through quantization is paved with both opportunities and challenges. The findings highlight the significance of considering model storage formats and the actual storage reduction capabilities when implementing quantization.

We emphasize the crucial role of training data volume in maintaining the model's accuracy levels post-quantization. As the field progresses, finding the right balance between model size, computational efficiency, and accuracy—especially in the context of available training data—will be essential for harnessing the full potential of quantized neural networks. By continuing to explore and innovate in this area, we can unlock new possibilities and achieve breakthroughs that will drive the future of neural network technology. Let us remain committed to pushing the boundaries, confident that our collective ingenuity and dedication will lead to remarkable advancements.

6. References

[1] H. Y. Wang, S. M. Ma, L. Dong, S. H. Huang, H. J. Wang, L. X. Ma, F. Yang, R. P. Wang, Y. Wu, F. R. Wei, “BitNet: Scaling 1-Bit Transformers for Large Language Models,” arXiv:2310.11453, 2023.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov. D. Weissenborn, X.H. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Hpulsby, "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv:2010.11929 ,2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need." In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010, 2017.

[4] L. J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. CoRR, 2016.

[5] H. Wang, S. Ma, S. H., L. D., W.H. Wang, Z.L. Peng, Y. Wu, P. Bajaj, S. Singhal, A. Benhaim, B. Patra, Z. Liu, V. Chaudhary, X. Song, and F.R Wei. Foundation transformers. CoRR, 2022.

[6] N.Rogge, "Transformers Tutorials (Version 1.0) [Computer software]". https://doi.org/10.5281/zenodo.1234 2020

[7] A. Krizhevsky, V. Nair, G. Hinton. “Learning Multiple Layers of Features from Tiny Images” 2009

[8] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition.” In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). 2016

[9] K. Simonyan, A. Zisserman “Very deep convolutional networks for large-scale image recognition”. *arXiv preprint arXiv:1409.1556*. 2014.

[10] S. Ma, H. Y. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, F. Wei, “The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits,” arXiv:2402.17764, 2024.