1 Bit ViT: Low-Bit Quantization Methods for Vision Transformers

Gautom Das **Nicholas Henderson**

Sean Im

Vincent La **Ethan Lau**

1 Introduction and Motivation

The efficient deployment of neural networks has always been an important problem. With more companies developing large language models (LLMs) on huge, costly GPU clusters, research into how to adapt these models to run on consumer devices is a growing field. One new popular method to increase the efficiency of these LLMs is known as "quantization". This method has traditionally been applied to classic language-based LLMs. The goal of this project is to apply low-bit quantization to vision transformers.

Transformers, originally introduced by Vaswani et al. (2017), revolutionized natural language processing (NLP) by enabling machines to better understand and generate text through handling sequential data. This success in NLP led researchers to adapt transformers for computer vision, resulting in the development of vision transformers (ViTs). ViTs, as described by Dosovitskiy et al. (2021), apply the transformer architecture directly to images by treating them as sequences of patches. This approach demonstrated an alternative to conventional convolutional neural networks (CNNs) that also achieves great results and can sometimes be more efficient.

Quantization is a method that reduces the precision of a network's parameters to make the model smaller and faster. For example, if you had a model where all the weights were 16-bit floating numbers, quantizing it to be 8-bit integers would roughly half the size of the model and also allow for faster calculations. This is especially useful on devices with limited memory or compute power. The concept of quantization, particularly for efficient integer-arithmetic-only inference, was initially developed by Jacob et al. (2018) at Google. This study found it was possible to convert complex floating-point based models into simple integer-only ones that could run on mobile devices. Since then more recent developments such as GPTQ and BitNet have further researched the potential for significant model compression. GPTQ, is a method that allows large GPT models to be compressed without a significant loss in performance, making them suitable for simpler hardware (Frantar et al., 2023).

The goal of this project is to apply low-bit quantization for language models to vision transformers. This could make vision transformers more energy-efficient and accessible for a wider range of devices, including those not traditionally equipped for advanced image recognition tasks. We take an existing vision transformer architecture, the ViT, with existing weights and attempt to quantize them using the methods prescribed in the original BitNet implementation and the 1.58 Bit method. Overall, this project can have applications in various sectors where efficient AI makes it more accessible to more users.

2 Background and Previous Works

2.1 4-bit Quantization

The foundational work on 4-bit quantization of language models provides valuable insights into maintaining accuracy under significant compression. This proposal envisions a similar leap for ViTs, utilizing the principles that have guided the successful quantization of language models. The goal is to achieve a balance where the efficiency gains from quantization do not come at the expense of the model's performance in image recognition tasks.

The "Towards Accurate Post-Training Quantization for Vision Transformer" study by Yifu Ding et al. is particularly instructive for our endeavor. It offers a novel approach to post-training quantization that significantly boosts ViTs' performance, especially in lower bit-width settings. This framework, with its innovative calibration scheme and attention to the Softmax function's integrity, serves as a blueprint for our project. It demonstrates the feasibility and benefits of applying low-bit quantization to ViTs, inspiring our approach to overcome the computational limitations we face.

2.2 1-bit and 1.58-bit Quantization

On the more extreme side of quantization is binarization, or quantization down to the size of a single bit. Recently, binarization has been applied to large language models, specifically the transformer architecture. BitNet, introduced by Wang et al. (2023) has shown that binarization can be a fruitful approach for large language models. Experimental results from BitNet show that a 1-bit architecture can achieve competitive performance while reducing memory footprint and energy consumption even more than that of 8-bit or 4-bit quantization.

The authors utilize group quantization and normalization, which divides weights and activations into groups and then estimates the parameters of each group independently. This allows their matrix multiplications to be partitioned across multiple devices since parameters can be computed locally, without further communication, which is important for scaling up large language models. Furthermore, the authors used mixed precision training, quanitizing weights and activations to a single bit but storing gradients and optimizer states in a high-precision format to ensure training stability and accuracy. For optimization, the authors suggest a large learning rate, since a small update often makes no difference for 1-bit weights. These findings will be important for us to consider, especially if we look to train a 1-bit ViT from scratch. A PyTorch implementation of BitNet has also been published including a `BitLinear` module in their code repository, intended as a drop-in replacement for a `nn.Linear` layer.

Variants of this 1-bit architecture are being researched such as BitNet b1.58, introduced by Ma et al. (2024). The authors here allow weights to take on 3 possible values -1,0,1 as opposed to just 2 possible values in BitNet. The authors have also found that Bitnet b1.58 retains a lot of the benefits of BitNet in terms of memory and energy consumption. An additional advantage is stronger modeling capability with the inclusion of 0 in the model weights which supports explicit support for feature filtering. Their work demonstrates a new scaling law with respect to model performance and inference cost between their quantized model and full precision models.

3 Methods

3.1 1-Bit Quantization

As highlighted in the previous works section, 1 bit quantization (or binarization) offers an extreme form of quantization where each weight in the model is represented with just one bit, taking values of either 0 or 1, or -1 and +1 in the case of symmetric quantization. This method offers significant reductions in memory requirements and increases in computational speed, especially during inference. The BitNet authors were able to show that despite the reduced bit-depth, BitNet achieves competitive results compared to models using more conventional 8-bit and FP16 formats. This approach not only conserves resources but also scales similarly to full-precision models, indicating its potential for future large-scale applications.

Below are the formulas we used to implement 1-bit quantization, taken from the BitNet paper:

The binarization of a weight $W \in \mathbb{R}^{n \times m}$ can be formulated as:

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

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

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

After initial binarization, we quantize the activations to b-bit precision. Then, we apply absmax quantization, scaling activations into the range $[-Q_b, Q_b]$ ($Q_b = 2^{b-1}$) by multiplying with Q_b and dividing by the absolute maximum of the input matrix:

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

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

where ϵ is a small floating point value that prevents overflow when clipping.

We scale the activations prior to the non-linear functions (e.g., RELU) into the range $[0, Q_b]$ by subtracting the minimum of the inputs to ensure all values are non-negative

$$\tilde{x} = \text{Quant}(x) = \text{Clip}\left((x - \eta) \times \frac{Q_b}{\gamma}, \epsilon, Q_b - \epsilon\right), \quad \eta = \min_{i,j} x_{ij}$$

3.2 1.58-bit Quantization

Moving beyond traditional 1-bit quantization, recent advancements have introduced the concept of 1.58-bit quantization. This approach uses a ternary weight system, where each parameter can take one of three possible values: -1, 0, or +1. The use of a ternary system allows for more nuanced control over model parameters compared to binary quantization, potentially offering a better balance between model size, speed, and accuracy. Below are the formulas we used to implement 1.58-bit quantization from Ma et .al (2024).

To start, we adopt an *absmean* quantization function to constrain the weights to -1, 0, or $+1$. The function first scales the weight matrix by its average absolute value, then rounding each value to the nearest integer among $\{-1, 0, +1\}$:

$$\widetilde{W} = \text{RoundClip}\left(\frac{W}{\gamma + \epsilon}, -1, 1\right), \tag{1}$$

$$\text{RoundClip}(x, a, b) = \max(a, \min(b, \text{round}(x))), \tag{2}$$

$$\gamma = \frac{1}{nm} \sum_{ij} |W_{ij}|. \tag{3}$$

The 1.58-bit quantization function follows the same implementation as BitNet, aside from the choice not to scale the activations prior to the non-linear functions to the range $[0, Q_b]$. Instead, the activations are scaled to $[-Q_b, Q_b]$ per token to avoid zero-point quantization. This is a simpler and more convenient approach for both implementation and optimization, while still only introducing negligible effects on performance .

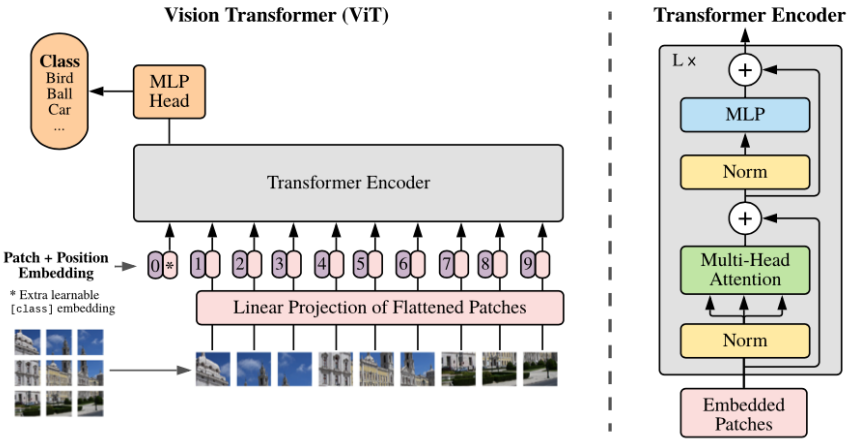


Figure 1: ViT architecture from "An Image is Worth 16x16 Words" paper.

3.3 BitLinear Class Implementation

To implement the 1-bit and 1.58-bit quantization techniques within the Transformer architecture, we introduced a new class, `BitLinear`, derived from PyTorch's `nn.Linear`. This class modifies the typical linear layer to support low-precision weights during both training and inference phases. The `BitLinear` layer implements a quantization-aware training approach, which integrates the quantization process directly into the training pipeline, thereby optimizing the model's performance for low-bit operations.

The `BitLinear` class replaces the weight matrix in the standard `nn.Linear` layer with a binarized version. This replacement is crucial for maintaining computational efficiency and adhering to the quantization schema proposed in BitNet. We utilized both the formulas highlighted in (1)-(5) and a 1-bit implementation repository authored by GitHub user kyegomez to help guide our experimentation.

3.4 Extending to 1.58 Bits

In order to extend the `BitLinear` class for 1.58-bit quantization we needed to adjust the quantization function to support ternary weights. This is achieved by scaling the weights by their average absolute value before applying a rounding function that maps them to one of the three values: -1, 0, or +1. As outlined in formulas (1)-(3).

The implementation for 1.58-bit quantization maintains the simplicity and efficiency of the BitLinear class while accommodating the additional complexity of ternary quantization. This enables the model to leverage the benefits of quantization, such as reduced memory and computational demands, with improved performance compared to binary (1-bit) quantization.

3.5 Original ViT Architecture

There are a number of `nn.Linear` layers that we can quantize in the Vision Transformer (ViT) architechture. ViT involves three major steps: patch embedding, transformer encoder, and classification head.

In the patch embedding stage, the input image is split into fixed-sized patches and flattened into numeric vectors called embeddings. CLS token and positional embedding is added as well. In this step, a single Conv2d layer is used instead of Linear for performance gains.

In the transformer stage, the ViT uses only the encoder part of the typical transformer architecture. As seen in the ViT architecture image, the transformer encoder has several stages: Norm, Multi-Head Attention, Residual Add / Connection, Norm, Multi-Layer Perceptron (MLP), and Residual Add.

Multi-Head Attention computes the attention matrix based on three inputs: queries, keys, and values. It uses multiple attention heads, meaning that multiple attentions are computed in parallel. MLP,

also known as the feedforward network, takes in the output of the attention and then upsamples and transforms it for the next layer. Norm is applied before every block, and residual add happens after every block. Norm layer normalizes the layer which improves performance of training. It is preferred over batch normalization as transformers operate on sequences of data. Residual add prevents vanishing gradient problem. These two layers exist for more of a technical reason.

Norm uses a `nn.LayerNorm` layer. Residual add does not make any torch.nn call. Multi-Head Attention uses 4 `nn.Linear` layers, one for each query, key, value, and out, and 2 additional `nn.Dropout` layers. MLP uses 2 `nn.Linear`, 1 `nn.GELU`, and 1 `nn.Dropout`. In total, that is 6 `nn.Linear` layers.

In the classification head stage, one final `nn.Linear` layer is used. We chose to keep this layer in full-precision because empirically the quantized models struggled to train when we quantized this final layer. Leaving it in full-precision yielded more stable training and overall better performance on test sets. Exploration of using mixed-precision within other parts of the ViT architechture is worth exploring in future work.

3.6 Implementing Quantized ViT

Our implementation of a quantized ViT adapts code from [6], which is a Pytorch re-implementation of ViT from [3]. We then wrote custom pytorch modules, adapted from [9] and [12], to replace `nn.linear` layers in a similar fashion to the original BitNet paper. We also employ mixed-precision training. These `BitLinear` layers quantize the latent weights (which are stored in full-precision) on the fly during training to either 1-bit or 1.58-bit precision. Activations were quantized to 8-bit precision but gradients and optimzier states were kept in high-precision formats for training stability and accuracy, much like the original BitNet paper.

3.7 Finetune Training and Evaluation

We finetune Google's ViT-B_16-244 pre-trained model for classfication tasks on the CIFAR-10 and CIFAR-100 datasets at different levels of precision: 1-bit, 1.58-bit, and full precision. The CIFAR-10 data set consists of 60000, 32x32 RGB images in that belong to 10 classes —there are 600 images in class. The CIFAR-100 data also has 60000, 32x32 RGB images but instead contains 100 classes with 600 images belonging to each class. ViT-B_16-244 is pretrained on imagenet-21k and finetuned on imagenet2012. Ideally, we would have liked to have trained quantized ViTs from scratch but given the time constraints of this project and our limited access to GPUs, this was not feasible. The scope of our work explores the feasibility of low-bit ViTs, their difference in performance to full-precision models, and low-bit fine-tuning. We leave the topic of training low-bit precision ViTs from scratch for future work to explore.

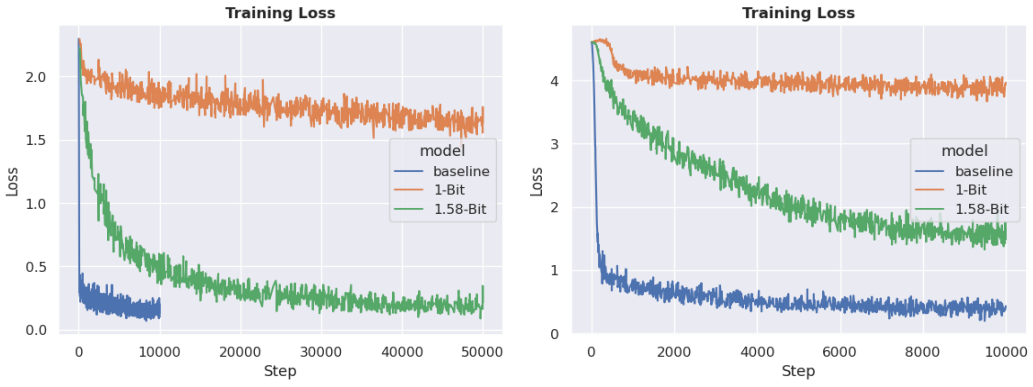
For the evaluation, the respective ViT model is instantiated with ViT-B_16 configuration, then the finetuned model weights are loaded to the instantiated ViT model. The models are then evaluated on their respective data sets (CIFAR-10, CIFAR-100) where we report the following metrics: accuracy, precision, recall, and F1 score. We also compute an overall confusion matrix and report this as a heatmap for visual representation of the model predictions across the test sets.

Another interesting way to evaluate a ViT is through a visual attention map. We compute the attention map by averaging the attention weights across all attention heads. We then add an identity matrix and re-normalize to account for residual connections. These matrices are multiplied against each other to obtain joint attention which is reshaped into the form of a mask image which is combined with the original image. The visual attention map highlights the parts of the image that the models attend to for our classification tasks.

4 Results

Our evaluations started with fine-tuning a base model ViT that was pre-trained on imagenet at full-precision on the CIFAR-10 and CIFAR-100 datasets. These training loss plots give us an understanding of how quickly and well the original architecture learns. The loss plots for the training process are in Figure 2. For CIFAR-10, we stopped training at around 10000 epochs as the model was beginning to over fit but achieved a loss of $\sim$0.2 on the dataset. For CIFAR-100, we also trained

to model to 10000 epochs and achieved a loss of ∼0.5. Once we have that baseline, we then fine tune the 1-bit and 1.58-bit models in a similar way. We will present the results for the 1-bit model first, followed by the 1.58-bit model, and finally a discussion of them both.



(a) (b)

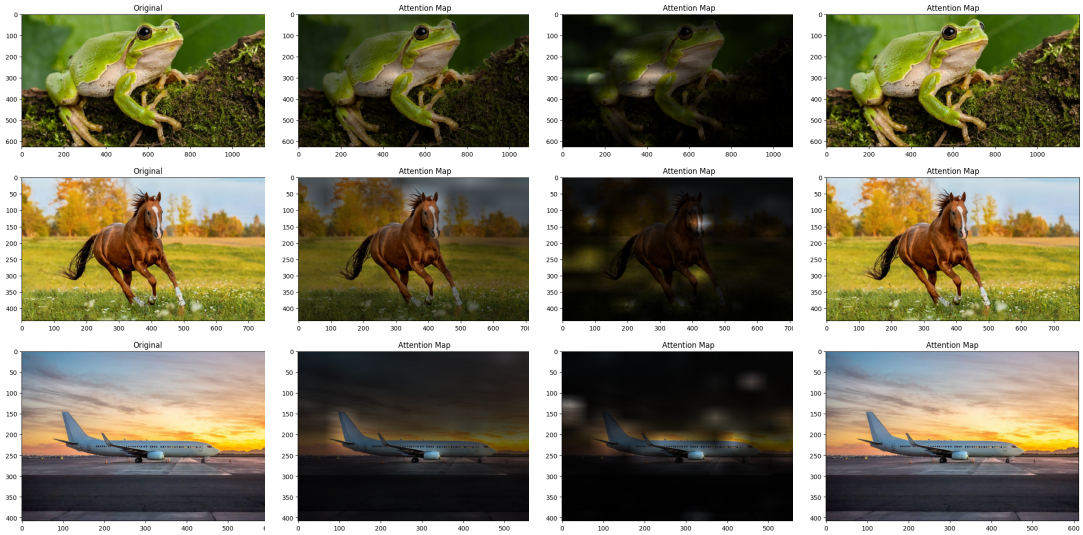
Figure 2: Loss Plots for (a) CIFAR-10 and (b) CIFAR-100

4.1 1-Bit Model

From the training loss plots in Figure 2, we can see how the 1-bit model struggled to generalize over both datasets. This is further seen when we test the model the 1-bit model on the unset test data in Table 1 and 2. For CIFAR-10, the model only achieved an accuracy of 43% compared to the baseline accuracy of 99%. For CIFAR-100, the model did even worse only having an accuracy of 10% compared to the baseline model's accuracy of 92%. The pretrained model seems to lose a lot of the general information learned about images when we binarize the weights. From the heatmaps in Figure 3, we can see that unlike the baseline model in column (b), the 1-bit attention model doesn't have notable attention regions. Finally, from the confusion matrices in Figure 4, we can see how the 1-bit model still has a lot of confusion with various classes such as confusing similar animals like dogs, cats, and dears or vehicles like ships, trucks, and airplanes. In the CIFAR-100 confusion matrix in Figure 5, we get a much more global view of this confusion and see how there are more more scattered hotspots that are not as concentrated on the diagonal as the baseline model. Overall, while the 1-bit model does demonstrate the ability to capture features for a low number of classes, we find this doesn't scale well as the number of classes increases.

4.2 1.58-Bit Model

The 1.58-bit model performed much better on both the CIFAR-10 and CIFAR-100 datasets. From the training loss plots in Figure 2, we can see how the 1.58-bit model was able to achieve much lower losses than the 1-bit model. For CIFAR-10, the 1.58-bit model achieves a loss of ∼0.25, much closer to the baseline model's loss of ∼0.2. For CIFAR-100, we obtained a final training loss of ∼1.6 which is not as close to the baseline value of ∼0.5 but much better than the 1-bit model's final loss of ∼3.9. Moving onto the test datasets we can see that on CIFAR-10, the 1.58-bit model has impressive accuracies of 95% compared to the baseline of 99%. However for CIFAR-100, this accuracy now drops to 66% compared to the baseline accuracy of 92%. To understand why this accuracy drops off as the number of classes we can again look to confusion matrices in Figure 4 and 5. For CIFAR-10, we can see while there is some minor confusion between some classes such as animals and vehicles, the model accurately predicts classes the majority of the time. For CIFAR-100, we can now see how this diagonal is no longer as densely concentrated and there are varying hotspots and confusion with random classes. While, not as severe as the 1-bit model it is still notable. Finally, for the 1.58-bit model's attention map we have much more interesting results compared to the baseline model. In each image example —most notaby the frog and the plane— the baseline model's attention map appears to highlight the subject of interest. For the 1.58-bit model, instead of surrounding the subject, the attention map highlights specific attributes such as the belly of the frog or the engine of the plane



(a) (b) (c) (d)

Figure 3: Attention maps of example images. (a) Original, (b) Baseline attention, (c) 1.58-bit attention, (d) 1-bit attention.

as the most significant feature to pay attention to. This suggests important implications discussed in the following section

Table 1: CIFAR-10 Test Results for Different Models

Metric	Baseline	1-Bit (50,000 Epochs)	1.58-Bit (50,000 Epochs)
Accuracy	0.9904	0.4361	0.9581
Precision	0.9904	0.4309	0.9583
Recall	0.9904	0.4361	0.9581
F1 Score	0.9904	0.4335	0.9582

Table 2: CIFAR-100 Test Results for Different Models

Metric	Baseline	1-Bit (10,000 Epochs)	1.58-Bit (10,000 Epochs)
Accuracy	0.9279	0.1096	0.6611
Precision	0.9286	0.0829	0.6640
Recall	0.9279	0.1096	0.6611
F1 Score	0.9282	0.0944	0.6625

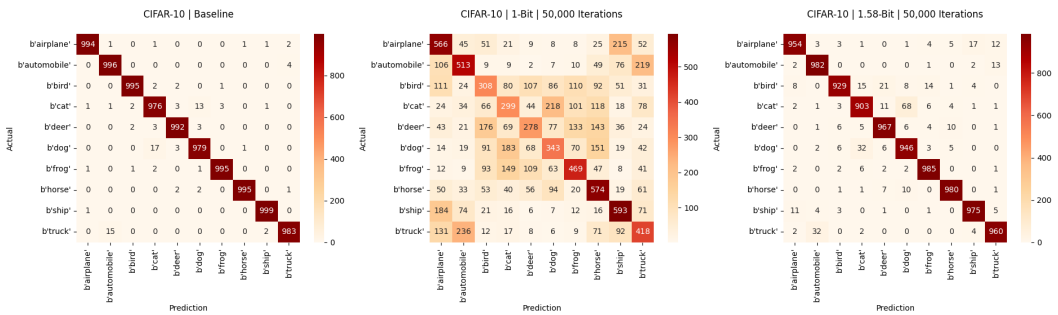


Figure 4: Confusion Matrices for CIFAR-10 Classification Task

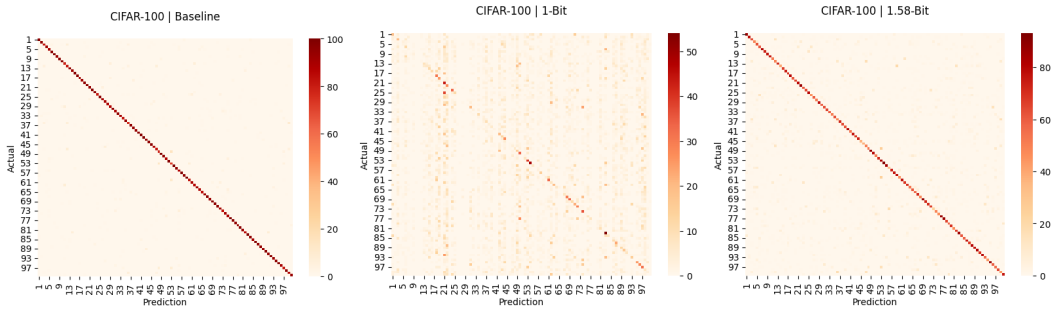


Figure 5: Confusion Matrices for CIFAR-100 Classification Task

4.3 Discussion

This study demonstrates the feasibility of applying extreme low-bit quantization techniques, specifically 1-bit and 1.58-bit, to vision transformers, a domain previously unexplored with such low quantization levels. Prior research had only studied quantization in ViTs to a minimum of 6 bits (Liu et al. 2024), making our exploration into sub-2-bit quantization particularly novel. Our findings reveal a clear relationship between quantization depth and model performance, specifically in image classification tasks. The 1-bit model, while showcasing the potential for substantial memory and computational efficiency, struggled significantly in maintaining accuracy, particularly as the number of classes increased. This was evident from the model's performance on the CIFAR-100 dataset, where it only managed 10% accuracy. On the other hand, the 1.58-bit model displayed a more robust capability, retaining much of the baseline model's performance, indicating that a ternary quantization approach could offer a viable trade-off between efficiency and accuracy. During the course of this project, we encountered several technical challenges, particularly with integration into existing quantization frameworks, which are not optimized for such low-bit operations. Moreover, current hardware limitations mean that the benefits of reduced data precision do not directly translate into speed improvements, as most GPU architectures are still only optimized for at least 8-bit operations.

5 Conclusion

In conclusion, this study successfully extends the realm of low-bit quantization to vision transformers, exploring quantization depths of 1-bit and 1.58-bit. These findings highlight a promising balance between computational efficiency and model performance, particularly with the 1.58-bit quantization which retains substantial accuracy compared to the baseline model. This research paves the way for further exploration into ultra-low-bit quantization in more complex models and diverse applications. Future studies could explore the dynamics between quantization depth, model complexity, and performance, potentially revolutionizing the deployment of advanced models on resource-constrained devices. Moreover, evaluating the impact of such quantization on power consumption in specialized hardware could validate the practical benefits of this approach, offering significant advancements in making high-performing, efficient AI technologies more accessible.

6 References

[1] Artidoro. (2023, July 19). Artidoro/Qlora: Qlora: Efficient finetuning of quantized llms. GitHub. https://github.com/artidoro/qlora

[2] Ding, Y., Qin, H., Yan, Q., Chai, Z., Liu, J., Wei, X., Liu, X. (2023, March 25). Towards accurate post-training quantization for vision transformer. arXiv.org. https://arxiv.org/abs/2303.14341

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929.

[4] Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D. (2023). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv preprint arXiv:2301.12345.

[5] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. arXiv preprint arXiv:1712.05877.

[6] Jeonsworld. (n.d.). Jeonsworld/VIT-pytorch: Pytorch reimplementation of the Vision Transformer (an image is worth 16x16 words: Transformers for image recognition at scale). GitHub. https://github.com/jeonsworld/ViT-pytorch

[7] Liu, Z., Wang, Y., Han, K., Ma, S., Gao, W. (2021). Post-Training Quantization for Vision Transformer. https://arxiv.org/abs/2106.14156

[8] Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., ... Wei, F. (2024). The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2402.17764

[9] Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., ... Wei, F. (2023). BitNet: Scaling 1-bit Transformers for Large Language Models. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2310.11453

[10] Wang, H. (n.d.). BitNet. GitHub. https://github.com/kyegomez/BitNet

[11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017).

[12] Wu, X., Li, C., Reza Yazdani Aminabadi, Yao, Z., He, Y. (2023). Understanding INT4 Quantization for Transformer Models: Latency Speedup, Composability, and Failure Cases. https://doi.org/10.48550/arxiv.2301.12017

[13]`1bitLLM/bitnet_b1_58-xl` Hugging Face. (n.d.). Hugging Face. `https://huggingface.co/1bitLLM/bitnet_b1_58-xl`