Initial Research Plan

Taowen(Tony) Liu

January 2024

1 Abstract

2 Introduction

This research proposal focuses on the numerical precision in machine learning (ML) training processes. Our objectives are: Firstly, to dissect the influence of numerical representation on ML training dynamics; secondly, to construct an accessible, precision-optimized ML library to facilitate research efficiency.

Objective 1: Establishing a Theoratical Framework for Numerical Precision's Role in ML Training Dynamics

We aim to understand the parameters with which arithmetic precision can be minimized without harm to ML model performance. This involves identifying specific conditions where reduced precision is viable. Additionally, we propose to establish a metric that correlates the degree of information loss due to number representation with ML training performance.

Objective 2: Integrating Precision Optimization Theories into ML Training Architectures

We aim to apply our theoretical knowledge from three aspects of ML training: computation, memory efficiency, and distributed training environments. At the computational level, we aim to influence the design of next-generation ML accelerators with optimized number formats. For memory usage, our objective is to enhance current hardware's efficiency, minimizing memory costs in both pre-training and fine-tuning scenarios. Lastly, we seek to find alternative numerical formats for distributed training settings or co-design the number representation and optimisers.

This passage also outlines our approach to assessing the impact of low precision arithmetic on ML training. This passage start by examining the latest developments in low precision number formats. We categorize ML workloads to understand their demands in the context of low precision computation.

In this plan we will analyze how low precision arithmetic affects different ML workloads, identifying performance trade-offs and documenting challenges and gaps in current research.

3 Background

Tell a story about the background

In the background part, we review number format and various machine learning training workload that we aim to optimise.

3.1 Number format

some text explain the overall content related to number format

3.1.1 Stochastic Rounding

In this research, we would like to explore utilising an alternative rounding scheme, stochastic rounding [1, 2], in machine learning training workload. To describe this method, let us consider a given number $x \notin F$ and two adjacent numbers x_1 and x_2 in F, where $x_1 < x < x_2$. The probability of rounding to x_1 or x_2 is proportional to one minus the relative distance from x to each number. In this scenario, we round up to x_2 with a probability of $\frac{x-x_1}{x_2-x_1}$ and down to x_1 with a probability of $\frac{x_2-x}{x_2-x_1}$.

Stochastic Rounding [1] can be implemented by examining residuals and utilizing a pseudorandom number generator (PRNG).

3.1.2 Floating Point Numbers

The IEEE Floating Point uses a sign bit, an exponent with a fixed bias, and a mantissa. It is characterized by the formula:

$$\text{Value} = (-1)^{\text{sign}} \times (1.\text{mantissa})_{\text{binary}} \times 2^{(\text{exponent}-\text{bias})}$$

This format includes special cases for subnormal numbers, infinity (Inf), and not-a-number (NaN). IEEE standard defines .

Minifloat is a mini version of the standard floating point format. It utilizes fewer bits for both the exponent and the mantissa, which leads to a reduced range and precision. Typically, minifloats do not include representations for Inf and NaN, but they usually support subnormal numbers.

3.1.3 Block Floating Point Numbers

Block mini float [3] utilise shared exponent bias across number blocks, enhancing dynamic range and accuracy. This format is especially efficient for arrays of numbers with similar magnitudes, enabling training with fewer exponent bits, because it is possible to implement kulish accumulator for FMA operation.

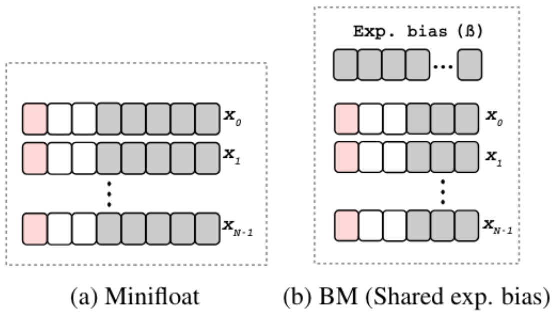

Figure 1: Minifloat and Block Minifloat (BM) tensor representations

Shared microexponents (MX) [4] is a framework for shared exponent bias formats in deep learning. MX employs scaling with a configurable scaling factors, improving quantization accuracy and efficiency over traditional methods. The shared exponent part can be either int or floating point made it more flexible than the BM number formats.

3.2 Machine Learning Training

In building a large machine learning model [5], several key phases are typically involved. Initially, the focus is on exploring various model architectures and potential applications through network and task research. Once a preliminary model and target tasks are established, the model is scaled up and pretrained on a large dataset. Following pretraining, the model is fine-tuned for downstream tasks. The final stage is deployment, which includes model compression techniques such as quantization-aware training and model pruning.

Low precision training can reduce costs during the pretraining stage. It also lowers the barriers to fine-tuning models for downstream tasks by reducing both memory and computational resource requirements. Furthermore, knowledge gained from low precision training can be used for enhancing quantization-aware training strategies.

3.2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) [6] is an optimization algorithm in machine learning. The update rule of SGD for a parameter θ at iteration t is given by:

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t, x^{(i)}),$$

where η is the learning rate, $\nabla f(\theta_t, x^{(i)})$ is the gradient of the loss function f with respect to the parameter θ at iteration t, computed for a randomly selected data point (or mini-batch) $x^{(i)}$.

Mini-batch gradient descent use a batch of samples to improve the gradient estimation. Adam [7], further improves SGD algorithm by using adaptive learning rate for each parameter, which is the most common optimiser used in machine learning practice.

3.2.2 Quantization-Aware Training

Quantization-aware training [8, 9] typically begins with training a machine learning model at full precision. Following this, weight updates are conducted using a method known as Straight-Through Estimator (STE) [8]. This involves calculating weight gradients in high precision and maintaining a high precision copy of the weights. During a training iteration, weights (and optionally activations) are quantized. The gradients are then computed based on these low precision values but updated in high precision. This process can be represented as:

$$\begin{aligned}
\Delta W_{hp} &= \text{ComputeGradient}(W_{lp}, A_{lp}),\\
W_{hp} &= W_{hp} - \eta \Delta W_{hp},\\
W_{lp} &= \text{Quantize}(W_{hp}),
\end{aligned}$$

3.2.3 Distributed Training

NO REFERENCE IN THIS PART!!!

Distributed training can handle large-scale neural networks, improving training speed and acheiving privacy requirements. One of the challenges in distributed training is the communication overhead associated with the communication of model updates, gradients, between the nodes. In federated learning scenarios with non-iid data, optimizing communication while ensuring data privacy and model accuracy is vital.

3.2.4 Parameter Efficient Finetuning

Parameter efficient finetuning [10, 11, 12] addresses the computational and memory challenges which caused by the growing size of machine learning models, such as large language models (LLMs) and latent diffusion models. It uses strategies like adapters [10] which small modules inserted between pre-trained model layers, with only the adapters' parameters being trainable, to reduce the memory footprint during finetuning. Prompt tuning [12] is another strategy, adjusting model inputs with specific prompts to guide output generation with minimal parameter updates.

4 Previous Work and Limitations

4.1 Low Precision Training

In order to address the increasing demand for computation and memory, Mixed Precision Training [13] presents a method for the training of deep neural networks that employs half-precision floating point numbers for the storage of weights, activations, and gradients. To reduce effects of information loss due to the dynamic range, the authors utilised a single-precision copy of the weights and also loss scaling. The implementation of half-precision training reduce the memory usage of deep learning models by nearly 50%, while also achieved acceleration through the adoption of half-precision hardware units. This is the initial work utilise alternative number format than IEEE single precision numbers.

"WAGE" quantization [14] introduced an new approach that quantise weights, activations, gradients, and errors into low-bitwidth integers for both training and inference phases. This work also replaces batch normalization by a constant scaling layer for pure integer dataflow. However, integer representation caused a performance degradation in training loss and network performance. Also, special hardware is required for implementing different bit-width for WAGE values.

Fp8 training [15] introduces a scaling method for linear layers in LLMs. It utilise per-tensor scaling for weights, gradients, and activations using the absolute maximum. This approach prevents performance degradation by continuously updating scaling biases before each GEMM operation. However, the scaling operation is expensive in training, because it consists a reduction operation for finding the scaling ratio, and a elementwise operation for applying this scaling.

Stable and low precision training for large scale vision language model [16] introduce SwitchBack, a linear layer optimized for int8 quantized training. Traditional low-precision training methods that need fp8 units which is absent in most machine learning accelerators. SwitchBack employs int8 computations for in the forward pass and gradient estimation of activations, while using bf16 for weight gradient estimation. Furthermore, the study use StableAdamW,to correct the debiasing moving average in the Adam optimizer. StableAdamW contributes to stabilizing the training process.

4.2 Low Bit Optimiser

8-bit Optimizers use 8-bit optimiser state without harming the performance. The authors developed block-wise quantization for gradient variance estimation to reduce the quantisation error. This method divides input tensors into smaller blocks that is associated with a scaling factor. Apart from that the author discovered that embedding layers for NLP tasks would favor a higher precision for optimiser state for training stability.

4.3 Low Bit Parameter Efficient Finetuning

qlora [17]

5 Objectives

5.1 Establishing a Theoretical Framework for Numerical Precision's Role in ML Training Dynamics

All previous studies have primarily relied on empirical experiments to explore the effects of noise introduced by rounding errors on machine learning training dynamics. This approach leaves a theoretical gap in comprehensively understanding these impacts. Stochastic rounding is a valuable tool for error analysis, allowing for probabilistic rather than worst-case error analysis. Given the complexity of analyzing an entire neural network, a pragmatic approach would involve developing theories at a smaller scale first.

Our research aims to conduct an ablation study to pinpoint the aspects of arithmetic operations in ML training that are particularly sensitive to numerical precision. Conversely, we aim to identify operations where precision can be reduced with minimal impact on the training loss.

Furthermore, we intend to identify and validate metrics that can measure the extent of information loss due to low precision number representation and its correlation with ML training performance. The signal-to-noise ratio (SNR) serves as a potential metric to assess the fidelity of number formats in representing distribution. However, the efficacy of SNR or any other metric in elucidating the influence of low precision numbers on ML training dynamics remains uncertain. Our objective includes evaluating the suitability of these metrics in providing meaningful insights into the training process under conditions of reduced numerical precision.

5.2 Integrating Precision Optimization Theories into ML Training Architectures

Current-generation machine learning ASICs accelerators are often limited by a fixed set of arithmetic units, constraining the exploration of lower precision in ML training. Our focus is on three key aspects: guiding arithmetic unit design for future ML ASICs, overcoming memory limitations in current hardware by utilizing low precision numbers for storage efficiency, and co-design low bit optimizers and distributed learning algorithms alongside number formats.

5.2.1 Hardware Improvment

Proper scaling is essential for reduing noise hence enhancing training results. We propose employing block quantization and the MXINT family of number representations as superior scaling techniques to tensor-level quantization into hardware design.

5.2.2 Trade-Offs under Memory Constraints

Memory constraints significantly impact the training of neural networks. Introducing number formats as a new dimension offers opportunities in design space exploration. For example, both batch size and bit width influence the accuracy of gradient estimation, and reducing either can decrease the memory requirements for training. This presents a natural trade-off between batch size and bit width during training. Furthermore, an alternative strategy involves balancing between low precision recomputation and the caching of activation values to manage memory more efficiently.

5.2.3 Co-design optimizers and distributed learning

Low bit optimisers and distrbuted learning in low bit number representation fields are less studied fields. We would like to explore the potential of developing optimiser or distributed learning algorithm that is specifically adapted to low bit arithmetic environments.

6 Plan

- Imediate thing that I am doing right now

plan : find a metric

in order to

- Longer term objectives

To enhance our understanding of the underlying theory, we need to first develop tools that facilitate comprehensive exploration. Our goal is to create a visualization tool designed to monitor the distribution of numbers during the training process. Additionally, we aim to analyze memory usage across various number formats at different stages of machine learning training.

Before ESA, we would like to identify a metric that evaluates the impact of number format on the dynamics of training. We would also like to deliver some training library using low precision number to reduce memory usage.

References

[1] Mantas Mikaitis. Stochastic rounding: Algorithms and hardware accelerator.

[2] Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3):211631, 2022.

[3] Ayon Basumallik, Darius Bunandar, Nicholas Dronen, Nicholas Harris, Ludmila Levkova, Calvin McCarter, Lakshmi Nair, David Walter, and David Widemann. Adaptive block floating-point for analog deep learning hardware.

[4] Bita Darvish Rouhani, Ritchie Zhao, Venmugil Elango, Rasoul Shafipour, Mathew Hall, Maral Mesmakhosroshahi, Ankit More, Levi Melnick, Maximilian Golub, Girish Varatkar, et al. With shared microexponents, a little shifting goes a long way. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–13, 2023.

[5] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[6] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016.

[9] Shien Zhu, Luan HK Duong, and Weichen Liu. Xor-net: An efficient computation pipeline for binary neural network inference on edge devices. In *2020 IEEE 26th international conference on parallel and distributed systems (ICPADS)*, pages 124–131. IEEE, 2020.

[10] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*, 2020.

[11] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[12] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[13] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training.

[14] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks.

[15] Sergio P Perez, Yan Zhang, James Briggs, Charlie Blake, Josh Levy-Kramer, Paul Balanca, Carlo Luschi, Stephen Barlow, and Andrew William Fitzgibbon. Training and inference of large language models using 8-bit floating point. *arXiv preprint arXiv:2309.17224*, 2023.

[16] Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. Stable and low-precision training for large-scale vision-language models.

[17] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.