Article

Not peer-reviewed version

Learning with Fewer Bits Across Layers and Time in the Training of Foundation-Scale Transformers

Oliver Hartley , Priya Desai , Nathaniel Brooks , Eleanor Hughes , Beverley Marion *

Posted Date: 22 September 2025

doi: 10.20944/preprints202509.0719.v1

Keywords: low-precision training; quantization; foundation models; mixed-precision optimization; large-scale deep learning; efficient AI; numerical stability; transformer architectures; bit-level computation; energy-efficient machine learning

Article

Learning with Fewer Bits Across Layers and Time in the Training of Foundation-Scale Transformers

Oliver Hartley [1,2], Priya Desai [3], Nathaniel Brooks [2], Eleanor Hughes [3] and Beverley Marion [1,*]

1 School of Computer Science, Lancaster University
2 Department of Engineering Science, University of Oxford
3 Department of Computing, Imperial College London
* Correspondence: b.marion@lancaster.ac.uk

Abstract

The unprecedented scale of contemporary foundation models has catalyzed a dramatic shift in both the capabilities and the computational demands of modern machine learning systems. While the performance benefits of large-scale architectures such as transformers are well-documented across a wide spectrum of domains—including natural language processing, computer vision, code synthesis, and multimodal reasoning—their resource consumption during training and deployment poses increasingly critical challenges. In response to these constraints, low-precision arithmetic has emerged not merely as a hardware optimization, but as a central algorithmic and architectural consideration for building scalable, sustainable, and accessible AI systems. In this work, we examine the frontier of low-precision training for large-scale neural networks, with a focus on how quantized representations, reduced numerical formats, and precision-aware optimizers interact with the unique demands of training foundation models. We explore how bit-level reductions in forward and backward computation affect convergence, stability, and generalization, particularly in the context of transformer-based architectures that dominate today's state-of-the-art. Beyond empirical performance, we consider the theoretical and practical implications of quantized gradients, loss surface discretization, and the trade-offs introduced by aggressive precision constraints. Our analysis covers a broad range of methods, including mixed-precision training, dynamic loss scaling, 8-bit and 4-bit optimizer variants, quantization-aware initialization, and the role of master weights in mitigating numerical instability. We further discuss how precision can be dynamically allocated across layers and training phases, revealing new opportunities for adaptive learning systems that optimize both accuracy and efficiency. Finally, we address the broader system-level and ethical dimensions of low-precision training—ranging from hardware-software co-design and compiler-level integration to issues of robustness, fairness, and carbon footprint. By synthesizing these diverse threads, we argue that low-precision training represents a fundamental rethinking of the numerical foundations of deep learning, one that will be essential for the next generation of AI models that are not only larger and faster, but also more efficient, equitable, and environmentally viable.

Keywords: low-precision training; quantization; foundation models; mixed-precision optimization; large-scale deep learning; efficient AI; numerical stability; transformer architectures; bit-level computation; energy-efficient machine learning

1. Introduction

The recent explosion of large foundation models, including but not limited to GPT-style transformers, vision-language models, and multimodal architectures, has catalyzed a fundamental transformation in artificial intelligence (AI) and machine learning (ML). These models, often comprising hundreds of billions to trillions of parameters, have demonstrated remarkable capabilities across a diverse spectrum of tasks, ranging from natural language understanding and generation to image recognition, translation, code synthesis, and beyond. However, the unprecedented scale of these models brings forth immense computational and memory challenges that severely limit their accessibility, trainability, and deployability, particularly in resource-constrained environments. One of the most promising directions to address these issues is the adoption of *low-precision training* methods—techniques that leverage reduced numerical precision to mitigate the prohibitive costs of training and inference [1]. Traditionally, training deep neural networks has relied on 32-bit floating point (FP32) arithmetic to ensure numerical stability and convergence guarantees [2]. However, this choice is no longer tenable as model sizes grow exponentially [3]. The use of lower-precision formats such as 16-bit (FP16, bfloat16), 8-bit (INT8, FP8), and even sub-8-bit representations (e.g., 4-bit and ternary formats) has emerged as a viable strategy to reduce the memory footprint, increase computational throughput, and enhance energy efficiency without significantly compromising model accuracy. These advances are not merely hardware-level optimizations but also involve a sophisticated interplay of numerical analysis, algorithmic design, hardware-software co-optimization, and theoretical understanding of gradient dynamics under quantization. In recent years, the research community has proposed a wide array of methods and frameworks for enabling low-precision training of large foundation models [4]. These include quantization-aware training (QAT), post-training quantization (PTQ), mixed-precision training, quantized backpropagation, and novel data formats (e.g., NVIDIA's TensorFloat-32 and FP8 formats). Moreover, new gradient scaling techniques, adaptive rounding strategies, quantization-aware optimizers, and calibration heuristics have emerged to bridge the performance gap between low-precision and full-precision training [5]. Simultaneously, there is a surge in the development of specialized hardware accelerators—GPUs, TPUs, NPUs, and custom ASICs—that are tailored to support low-precision computation at scale [6]. Nevertheless, the deployment of low-precision training techniques to large foundation models introduces several open challenges [7]. Numerical instability, gradient underflow or overflow, loss of representational capacity, and compatibility issues with existing software frameworks and hardware backends can all impair the fidelity and convergence of training [8]. Moreover, training dynamics in extremely low-bit regimes (e.g., 4-bit or binary training) remain poorly understood, particularly in the context of transformer-based architectures with complex attention mechanisms and residual pathways. This necessitates a systematic and comprehensive exploration of the theoretical foundations, empirical performance, and practical implementations of low-precision training methods. The goal of this survey is to provide an exhaustive and structured overview of the field of low-precision training for large foundation models. We begin by tracing the historical evolution of precision formats in deep learning and situating low-precision training within the broader context of model efficiency. We then delve into the mathematical and algorithmic foundations of quantization, including quantization theory, error analysis, and the impact of precision on gradient flow and optimization dynamics [9]. Following this, we categorize and critically analyze existing methods for low-precision training across different model architectures, precision regimes, and training setups. Particular emphasis is placed on large-scale empirical results, case studies from industry and academia, and benchmarking efforts that highlight the current state-of-the-art. Furthermore, we explore the hardware and system-level implications of low-precision training, discussing how architectural innovations in processors and memory hierarchies intersect with algorithmic advances. This includes insights into compiler support, distributed training infrastructure, and real-world deployment pipelines. Finally, we outline pressing open problems, propose directions for future research, and examine the potential of ultra-low-precision regimes, neuromorphic computing, and hybrid precision approaches for pushing the boundaries of scalable AI. In sum, this survey serves as a comprehensive guide for researchers, engineers, and practitioners seeking to understand, utilize, and innovate in the domain of low-precision training for large foundation models. By consolidating theoretical insights, practical methodologies, and empirical evaluations, we aim to accelerate progress toward more efficient, accessible, and environmentally sustainable AI systems.

2. Background and Motivation

The training of large-scale foundation models involves the optimization of objective functions defined over extremely high-dimensional parameter spaces. Formally, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denote a dataset of N training examples, and let $f_\theta : \mathcal{X} \to \mathcal{Y}$ represent a neural network parameterized by $\theta \in \mathbb{R}^d$, where $d \gg 10^9$ in the case of modern large language models (LLMs) such as GPT-4 or PaLM. The training objective is typically to minimize the empirical risk:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(f_\theta(x_i), y_i),$$

where $\ell(\cdot,\cdot)$ is a suitable loss function, e.g., cross-entropy for classification or mean squared error for regression [10]. Optimization is performed via stochastic gradient descent (SGD) or its variants, updating parameters using:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla \mathcal{L}_{\mathcal{B}}(\theta_t),$$

where η is the learning rate and $\mathcal{L}_{\mathcal{B}}$ denotes the loss over a mini-batch $\mathcal{B} \subset \mathcal{D}$. However, when training in low-precision environments, all quantities—parameters θ, activations a_l, gradients $\nabla\theta$, and optimizer states—are represented using reduced numerical formats. Let $\mathcal{Q}_p : \mathbb{R} \to \mathbb{Q}_p$ denote a quantization function that maps real numbers to a set of representable values under a p-bit numerical format (e.g., $p = 16$ for FP16 or bfloat16, $p = 8$ for INT8 or FP8). Quantized training involves computing updates using:

$$\theta_{t+1} = \mathcal{Q}_p\big(\theta_t - \eta \cdot \mathcal{Q}_p(\nabla \mathcal{L}_{\mathcal{B}}(\theta_t))\big),$$

introducing quantization errors both in the forward and backward passes. These errors, while potentially small in isolation, can accumulate or interfere with the delicate gradient flow necessary for effective convergence in deep networks [11]. Specifically, quantization introduces both bias and variance into the stochastic gradients, potentially altering the optimization trajectory and leading to suboptimal solutions or training instability. The motivation for adopting low-precision training arises from the inherent trade-offs between numerical accuracy, compute efficiency, and energy consumption [12]. Given that the computational complexity of dense matrix-matrix multiplication scales as $\mathcal{O}(n^3)$ and memory complexity as $\mathcal{O}(n^2)$, where n denotes the size of activations or weight matrices in a given layer, any reduction in the bit-width of tensor representations yields quadratic or cubic savings in memory bandwidth and FLOPs. Moreover, modern accelerator hardware such as NVIDIA's Ampere and Hopper architectures, Google's TPU v4, and custom chips from Graphcore or Cerebras offer orders-of-magnitude higher throughput for low-precision tensor cores compared to standard FP32 units. Thus, operating in a low-precision regime becomes not just an optimization trick but a necessity for scaling to trillion-parameter models within practical time and energy budgets [13]. Another critical motivation stems from the environmental and economic cost of training large models [14]. Let E_p denote the energy consumption per operation in p-bit precision. It is empirically observed that $E_{16} \ll E_{32}$ and $E_8 \ll E_{16}$, with typical savings of $2\times$ to $4\times$ in power and up to $10\times$ in throughput [15]. When applied over millions of training steps and thousands of GPUs, the aggregate savings can amount to megawatt-hours of electricity and millions of dollars in compute cost. These factors are especially salient for organizations operating under carbon neutrality constraints or those deploying models in edge and mobile environments [16]. Despite the clear benefits, low-precision training introduces significant challenges. Key numerical issues include quantization-induced gradient vanishing, dynamic range limitations (e.g., due to exponent clipping), reduced representational fidelity in layer normalization and softmax operations, and incompatibility with standard optimizers that assume floating-point arithmetic (e.g., Adam or RMSProp) [17]. These issues necessitate the development of mixed-precision approaches, in which certain critical operations—e.g., accumulation in FP32 or parameter updates in higher precision—are selectively exempted from quantization [18]. Alternatively, adaptive precision strategies dynamically adjust bit-widths during training based on sensitivity metrics or signal-to-noise ratios. Furthermore, the theoretical underpinnings of low-precision optimization are still under active investigation [19]. While empirical results suggest that many models exhibit a degree of robustness to quantization noise, the precise relationship between quantization-induced perturbations and generalization performance remains poorly understood. Early theoretical efforts model quantization as additive or multiplicative noise, leading to modified stochastic differential equations governing the learning dynamics. Let $\epsilon_q \sim \mathcal{N}(0, \sigma_q^2)$ denote the quantization noise injected at each gradient step; then the update rule becomes:

$$\theta_{t+1} = \theta_t - \eta \cdot (\nabla \mathcal{L}_{\mathcal{B}}(\theta_t) + \epsilon_q),$$

which resembles a noisy gradient descent process. The magnitude and structure of σ_q^2 are determined by the quantization scheme, dynamic range, and precision level. Understanding how σ_q^2 interacts with the loss landscape is a critical step toward designing robust and theoretically grounded low-precision training algorithms. In summary, the motivation for low-precision training is driven by a confluence of computational, environmental, economic, and algorithmic imperatives [20]. It represents a compelling pathway to democratizing access to powerful foundation models, improving the sustainability of AI development, and enabling the deployment of intelligent systems at the edge. However, realizing its full potential demands rigorous mathematical modeling, robust engineering solutions, and a deeper understanding of precision-sensitive training dynamics [21]. The following sections will explore the full landscape of low-precision methods, categorizing them by architectural focus, precision regime, and theoretical foundation [22].

3. Precision Formats and Numerical Representations

At the heart of low-precision training lies the concept of reduced-bit numeric representations, which define how real-valued tensors (weights, activations, gradients) are encoded and manipulated during the learning process. These representations trade off numerical fidelity for efficiency in compute, memory, and energy. The choice of precision format directly impacts model performance, stability, and compatibility with hardware accelerators [23]. In this section, we review the most prominent low-precision formats used in deep learning and compare their structural properties in terms of bit allocation, dynamic range, quantization error, and hardware support [24].

Table 1. Overview of Numeric Formats for Low-Precision Training.

Format	Bits	Exp/Mant	Range	Type	Use Case	
FP32	32	8 / 23	$\sim 10^{\pm 38}$	Float	Full precision training	
FP16	16	5 / 10	$\sim 10^{\pm 5}$	Float	Mixed precision	
bfloat16	16	8 / 7	$\sim 10^{\pm 38}$	Float	TPU training	
TF32	19	8 / 10	$\sim 10^{\pm 38}$	Float	Tensor cores	
FP8 (E5M2)	8	5 / 2	$\sim 10^{\pm 6}$	Float	Experimental training	
INT8	8	– / 8	$[-128, 127]$	Fixed	Inference	
INT4	4	– / 4	$[-8, 7]$	Fixed	Edge inference	

Floating-point formats such as FP32 and FP16 conform to the IEEE-754 standard, with a sign bit, an exponent field controlling dynamic range, and a mantissa (or significand) determining precision. While FP32 remains the gold standard due to its wide dynamic range and fine-grained accuracy, it imposes significant compute and memory costs, making it suboptimal for large model training. FP16 halves the memory footprint and doubles throughput on compatible hardware, but suffers from a much narrower dynamic range due to its reduced exponent bits. This can lead to issues such as gradient underflow, particularly during backpropagation through deep networks or when using small batch sizes. To address FP16's limitations, the bfloat16 (brain float 16) format was introduced, especially favored by Google's TPU ecosystem. bfloat16 preserves FP32's exponent width while truncating the mantissa, maintaining a similar dynamic range at the cost of precision. This property makes it well-suited for numerically stable training of large models, especially when used with accumulation in higher precision. NVIDIA's TensorFloat-32 (TF32) format offers another compromise: it shares

FP32's exponent width and FP16's mantissa size, enabling fast matrix math on Ampere GPUs without the numerical instability of pure FP16 [25]. Lower still in the precision hierarchy, FP8 formats—such as E4M3 (4 exponent, 3 mantissa) and E5M2 (5 exponent, 2 mantissa)—are gaining traction for both training and inference [26]. Despite their severe quantization, FP8 formats are surprisingly effective when paired with techniques like quantization-aware training and stochastic rounding. Their compact size (just 1 byte per scalar) makes them attractive for high-throughput training of very large models. However, the extremely limited mantissa can introduce catastrophic rounding errors unless mitigated by careful scale management and forward/backward calibration. Fixed-point formats such as INT8 and INT4 represent another class of low-precision encodings where all bits are allocated to magnitude with an implicit scaling factor [27]. These are most effective when the distribution of values in weights or activations is known or can be calibrated in advance [28]. While fixed-point arithmetic is deterministic and hardware-friendly, it lacks the dynamic range flexibility of floating-point formats and can fail under high variance or sparse distributions. INT8 is already a de facto standard for inference on edge devices, and recent studies have demonstrated its viability for training as well—albeit with gradient accumulation in higher precision. The trade-offs among these formats are multi-dimensional. While reduced mantissa bits degrade precision and introduce quantization noise, reduced exponent bits narrow the dynamic range and can cause saturation or underflow [29]. Moreover, hardware support varies widely: GPUs like NVIDIA's A100 and H100 offer native FP16, TF32, and experimental FP8 support via tensor cores; TPUs excel at bfloat16; and CPUs are generally restricted to FP32 and INT8. These constraints necessitate format-aware algorithm design and hybrid strategies that select optimal precisions per tensor type, layer, or computation phase [30]. In conclusion, the landscape of numeric formats is both diverse and rapidly evolving. A deep understanding of each format's numerical characteristics, compatibility with training dynamics, and hardware implementation is essential for designing robust and efficient low-precision training pipelines [31]. In subsequent sections, we will examine how these formats are integrated into quantization algorithms, optimizer designs, and full-stack training frameworks [32].

4. Quantization Techniques for Training

Quantization is the process of mapping continuous real-valued tensors to discrete, low-precision representations, typically defined by a finite set of allowable numerical values. In the context of deep learning training, quantization can be applied to various entities: model weights, activations, gradients, and optimizer states. The challenge is to perform this mapping in a manner that preserves the essential characteristics of the data while ensuring the network remains trainable. Unlike inference quantization, which operates on frozen weights, training quantization must contend with dynamic updates, gradient noise, and non-stationary distributions [33].

As shown in Figure 1, a typical quantized training loop involves applying quantization functions $\mathcal{Q}_p$ to inputs before the forward pass and again to gradients during backpropagation. A key design choice is whether to perform all operations—forward propagation, gradient computation, and parameter updates—in low precision, or to retain high-precision accumulators for critical steps [34]. The most widely used practice today is *mixed-precision training*, where activations and weights are quantized to formats like FP16 or bfloat16, but gradients and parameter updates are stored in FP32 to prevent cumulative rounding errors. Quantization functions $\mathcal{Q}_p(\cdot)$ can take several forms. The simplest is uniform quantization, defined as:

$$\mathcal{Q}_p(x) = \text{clip}\left(\Delta \cdot \left\lfloor \frac{x}{\Delta} + 0.5 \right\rfloor, x_{\min}, x_{\max}\right),$$

where Δ is the quantization step size, and the clip function ensures values fall within the representable dynamic range. Non-uniform quantization schemes, such as logarithmic quantization or learned step-size quantization (LSQ), offer better representational fidelity for distributions with heavy tails or sharp peaks, such as those often seen in transformer attention scores or large-batch gradients [35]. Another key distinction is between symmetric and asymmetric quantization. Symmetric quantization assumes zero is the midpoint, simplifying hardware but limiting flexibility [36]. Asymmetric quantization uses separate scale and zero-point parameters, enabling better fit to data distributions at the cost of additional overhead. For instance, an asymmetric quantizer maps $x \in \mathbb{R}$ to:

$$\mathcal{Q}(x) = \left\lfloor \frac{x - z}{\Delta} \right\rfloor,$$

where z is the zero-point offset and Δ the scale [37]. In practice, quantization-aware training (QAT) is often preferred over post-training quantization (PTQ) for training large foundation models [38]. QAT introduces quantization operations during training itself, allowing the model to adapt its weights to compensate for quantization noise. This is especially useful in ultra-low-bit scenarios (e.g., 4-bit or binary networks), where naive quantization severely degrades performance. Techniques such as fake quantization—where quantized values are simulated during the forward pass but gradients are back-propagated through real-valued tensors—enable end-to-end differentiability despite discrete-valued intermediate computations. Moreover, the quantization of gradients poses particular challenges [39]. Gradients are typically much smaller in magnitude than activations or weights and are more susceptible to underflow in low-precision regimes. To counter this, gradient scaling is employed—either statically (using a fixed scale factor) or dynamically (adaptive per layer or per step). Formally, let g be the raw gradient, and s the scaling factor, then:

$$\hat{g} = \mathcal{Q}_p(s \cdot g)/s,$$

which rescales the quantized gradient back to its original scale while preserving the quantization error within tolerable bounds. In summary, quantization techniques for training large foundation models involve a complex but tractable balance between algorithmic fidelity and computational efficiency. Each design choice—precision format, quantization method, accumulation strategy—must consider both the mathematical properties of deep networks and the practical constraints of available hardware. As we explore further, we will delve into how these techniques are orchestrated at the optimizer level, integrated into full training frameworks, and generalized across various architectural backbones [40].

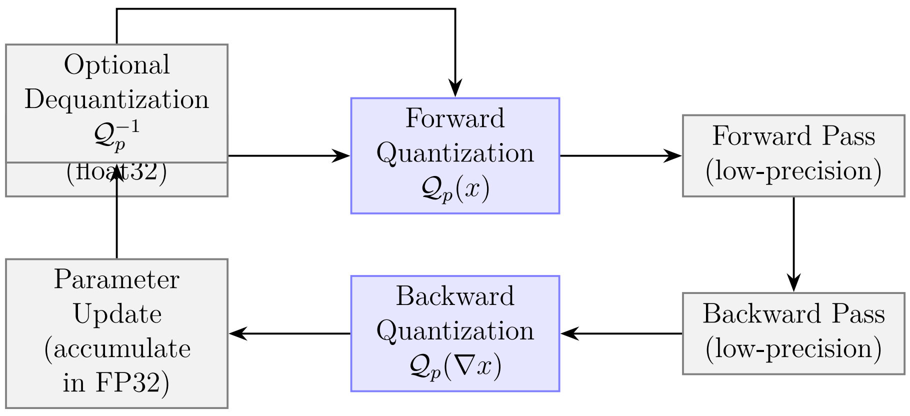
Optional Dequantization $\mathcal{Q}_p^{-1}$ (float32)
Forward Quantization $\mathcal{Q}_p(x)$
Forward Pass (low-precision)
Backward Pass (low-precision)
Backward Quantization $\mathcal{Q}_p(\nabla x)$
Parameter Update (accumulate in FP32)

Figure 1. Quantization pipeline during training. Forward and backward computations occur in low-precision. Updates are often accumulated in high-precision to preserve accuracy.

5. Optimizers and Gradient Handling in Low Precision

Optimizers play a central role in deep learning training, transforming raw gradient signals into effective parameter updates that ensure convergence and generalization. In full-precision settings, optimizers like SGD, Adam, and RMSProp rely on stable arithmetic with ample dynamic range and precision. However, in low-precision regimes, several nontrivial issues arise: gradients may underflow, moving averages may saturate, and moment estimates may lose fidelity due to quantization noise.

Consequently, optimizer design must be carefully reconsidered when training large models under precision constraints.

As shown in Table 2, modern training pipelines often rely on mixed-precision variants of adaptive optimizers. A common strategy is to perform all forward and backward passes in reduced precision (e.g., FP16 or bfloat16), while maintaining internal optimizer state variables—such as momentum and variance estimates—in full FP32 precision. This is essential for preventing the drift and decay instability that arises from rounding errors in iterative updates. For example, in the Adam optimizer, the first and second moments m_t and v_t are updated as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$$

where g_t is the gradient at time t. If m_t and v_t are stored in low-precision formats, repeated rounding and underflow may cause them to either explode or collapse to zero, leading to optimization failure. To mitigate this, implementations such as NVIDIA's Apex and PyTorch's native 'torch.cuda.amp' maintain FP32 copies of all optimizer states even when operating on quantized weights and activations. This approach is called "master weight storage," and it allows parameter updates to be computed accurately, even if the actual forward and backward passes use quantized tensors. Upon each update, the master weights are updated using the full-precision optimizer logic, and the quantized weights are refreshed accordingly. In more aggressive approaches, such as 8-bit optimizers (e.g., QAdam or 8-bit SGD), even the optimizer states are quantized using techniques like dynamic range scaling, learned rounding, and per-layer scale calibration [41]. These methods offer substantial memory and bandwidth savings, especially in multi-GPU or distributed setups. However, they are much more sensitive to initialization and require fine-tuned hyperparameters, including learning rate warmups and layer-wise clipping thresholds. For example, QAdam employs a dynamic quantization function $\mathcal{Q}_8(\cdot)$ for both moments and updates, with scale factors learned during training via backpropagation or heuristically estimated using moving windows. Gradient scaling is another indispensable component of low-precision training. Since many gradients in deep networks fall in the range $[10^{-6}, 10^{-3}]$, low-precision formats with limited exponent ranges (like FP16) cannot faithfully represent them [42]. To address this, a dynamic loss scaling factor γ_t is introduced to rescale all gradients before quantization:

$$\tilde{g}_t = \mathcal{Q}_p(\gamma_t \cdot g_t), \quad \text{then} \quad \hat{g}_t = \tilde{g}_t / \gamma_t.$$

Table 2. Optimizer Compatibility with Low-Precision Training.

Optimizer	LP Ready	FP32 Accum.	Dyn. Scaling
SGD (vanilla)	Partial	Optional	No
Momentum SGD	Yes	Yes	No
Adam	Yes	Yes	Yes
8-bit Adam (QAdam)	Yes	No	Yes
Adafactor	Yes	Yes	No

This technique preserves small gradient magnitudes while bounding overflow. The scale γ_t can be adjusted adaptively based on overflow detection or predefined heuristics, enabling both robustness and convergence. Advanced frameworks like DeepSpeed and FairScale provide support for such techniques out-of-the-box [43]. In conclusion, optimizer behavior in low-precision training must be explicitly managed to prevent divergence and instability. The use of mixed-precision accumulators, dynamic gradient scaling, and quantization-aware moment tracking constitutes a delicate balance between performance and stability [44]. Future work in this area may explore learned optimizer architectures that are natively precision-aware, or new optimization algorithms designed from first principles to operate in quantized numerical domains [45].

6. Hardware Support for Low-Precision Training

The proliferation of low-precision training methodologies is inextricably linked to the evolution of hardware architectures capable of supporting reduced-precision arithmetic efficiently and at scale. As the computational demands of foundation models have ballooned—reaching hundreds of billions or even trillions of parameters—conventional hardware and data center infrastructures have faced increasingly unsustainable pressures in terms of throughput, latency, power consumption, and cost [46]. In response, major hardware vendors have engineered accelerators specifically optimized for low-bit-width computation, including tensor cores, systolic arrays, and domain-specific instruction sets. These innovations are not merely auxiliary conveniences—they form the critical substrate upon which the feasibility of low-precision training depends. Consequently, understanding the architectural design principles, memory hierarchies, and execution models of contemporary low-precision hardware is essential for any serious engagement with the field. Modern GPUs, particularly those designed by NVIDIA, have spearheaded the adoption of reduced-precision computation in deep learning [47]. Starting with the Volta architecture, NVIDIA introduced Tensor Cores, specialized matrix multiplication units capable of performing mixed-precision operations with unprecedented efficiency. Subsequent architectures—Turing, Ampere, and Hopper—have successively extended support for a growing array of precision formats, including FP16, bfloat16, TF32, INT8, and most recently FP8. For instance, Ampere GPUs feature third-generation Tensor Cores that deliver up to 312 TFLOPS of FP16 performance and nearly 1000 TOPS (tera operations per second) for INT8 workloads. Hopper takes this further by introducing support for FP8 tensor math in hardware, enabling low-precision training without emulation overhead. The underlying architectural enhancement lies in the ability to perform fused multiply-accumulate (FMA) operations in low-precision input types while accumulating results in FP32, thus mitigating the catastrophic rounding errors that plague naïve low-bit computation. This "precision sandwiching" design philosophy enables high-throughput compute while preserving accuracy—an essential compromise for the demands of foundation model training [48]. On the other hand, Google's Tensor Processing Units (TPUs) offer a complementary view of hardware-software co-design. TPUs natively support bfloat16—a format that maintains FP32's 8-bit exponent but truncates the mantissa to just 7 bits. This design was explicitly chosen to balance numerical range with implementation simplicity [49]. Unlike NVIDIA's Tensor Cores, which prioritize wide format support and architectural flexibility, TPUs focus on throughput-optimized matrix operations executed on systolic arrays [50]. These arrays exploit spatial and temporal locality to achieve extremely high arithmetic density, allowing for efficient implementation of dense neural network layers [51]. With the introduction of TPU v4, Google has further refined its hardware stack, supporting petaflops of low-precision compute per chip and enabling distributed training across hundreds of pods. Importantly, Google's XLA compiler plays a key role in optimizing computation graphs for low-precision execution, including constant folding, memory tiling, and kernel fusion—transformations that are essential for leveraging the full power of TPUs in practical training scenarios. Beyond general-purpose accelerators, there is growing interest in domain-specific hardware for ultra-low-precision training [52]. Companies such as Graphcore, Cerebras, and SambaNova have introduced novel chip architectures that abandon traditional Von Neumann bottlenecks in favor of spatial computing, wafer-scale integration, and memory-in-logic designs. For instance, the Cerebras WSE-2 (Wafer-Scale Engine) integrates over 850,000 cores on a single silicon wafer, with local SRAM and mesh interconnects allowing for ultra-low-latency communication [53]. While such platforms are still relatively niche, they offer promising avenues for large-scale low-precision training due to their ability to co-locate computation and memory—thereby eliminating the energy cost of data movement, which can account for over 90% of power consumption in conventional GPU workloads. Similarly, Graphcore's IPU architecture emphasizes fine-grained parallelism and supports FP16 and mixed-precision instructions with flexible execution scheduling, making it suitable for dynamic graph workloads, such as those encountered in sparse transformers and recursive networks [54]. Memory subsystems also play a pivotal role in the overall performance and feasibility of low-precision training. Reduced precision formats, by definition, cut down the memory footprint of weights, activations, and gradients—allowing for larger batch sizes, deeper unrolling, or more layers to fit into the limited on-chip cache and HBM (High Bandwidth Memory) of modern accelerators [55]. However, the benefits of reduced memory bandwidth are only realized if the entire training pipeline is carefully co-optimized. Memory alignment, coalesced access patterns, and data layout transformations are all essential for minimizing latency and maximizing throughput. For example, NVIDIA's cuDNN and CUTLASS libraries provide primitives optimized for different precision modes, while Google's XLA and JAX pipelines include layout-aware optimization passes to reduce fragmentation and inefficient memory hops. As such, software abstractions and compiler toolchains have become integral components of the low-precision hardware ecosystem. From a systems perspective, distributed training frameworks must also be precision-aware to prevent bottlenecks in communication and synchronization [56]. Communication overhead in data-parallel training can dominate computation time when scaling to hundreds of GPUs. Techniques such as 8-bit gradient compression, quantized all-reduce, and low-precision collective operations (e.g., using NCCL's FP16 all-reduce support) are crucial for maintaining scalability in multi-node setups [57]. However, quantization of communication must be handled delicately to avoid degrading convergence; this often requires selective compression, error compensation buffers, and gradient clipping heuristics. Moreover, precision heterogeneity across hardware nodes—e.g., mixing GPUs with differing FP8/TF32/FP16 support—introduces additional complexity in pipeline design and synchronization semantics. In sum, hardware support for low-precision training is a multifaceted and rapidly evolving domain [58]. It encompasses not only the physical execution units capable of low-precision arithmetic, but also memory architecture, communication protocols, compiler toolchains, and software APIs that expose and abstract these capabilities to practitioners [59]. The full-stack optimization of low-precision training—from instruction sets to distributed systems—represents one of the most significant engineering efforts in modern AI infrastructure. As foundation models continue to scale and the demand for sustainable AI intensifies, the role of hardware co-design in enabling practical and performant low-precision training will only grow more central. Future directions may include the standardization of FP8 formats, hardware-software codesign for 4-bit training, integration of photonic or analog accelerators, and increased openness in vendor-specific instruction sets to promote cross-platform optimization.

7. Case Studies and Benchmarks

The theoretical appeal of low-precision training is only meaningful if it translates into empirical gains in real-world applications. Over the past few years, numerous case studies have demonstrated the viability of reduced-precision training across a broad spectrum of tasks and model scales, including image classification, natural language processing, code generation, and multimodal reasoning. These empirical evaluations not only validate the efficacy of various low-precision techniques but also reveal the nuanced trade-offs between speed, accuracy, memory savings, and stability. In this section, we examine a range of benchmark results and practical deployments to assess the performance of low-precision training in production-scale environments [60]. Importantly, we emphasize that success in low-precision regimes is not uniform across all architectures and datasets—it is highly sensitive to the underlying model dynamics, optimizer configurations, scale of training, and even seemingly minor engineering choices such as data prefetching, sharding, and initialization heuristics. One of the earliest large-scale demonstrations of mixed-precision training was presented by NVIDIA in their work on training ResNet-50 for ImageNet using FP16 arithmetic on V100 GPUs [61]. By leveraging Tensor Cores and dynamic loss scaling, they achieved over 2× speedup while matching the baseline FP32 top-1 accuracy of 76.3% within the same number of epochs. This result was later extended to deeper networks and larger datasets, including EfficientNet on ImageNet and SSD300 for object detection [62]. What is particularly striking is the consistency with which FP16-based training matched FP32 baselines once dynamic scaling, master weight storage, and properly tuned batch normalization were applied. These early successes laid the foundation for the widespread adoption of mixed-precision techniques in computer vision, where spatial locality and convolutional structure reduce the sensitivity to low-precision noise. Subsequent work by Google Brain, using bfloat16 on TPU v3, further confirmed this robustness, training large-scale Vision Transformers (ViTs) without sacrificing performance. Indeed, in the ViT-G/14 model trained on JFT-300M, bfloat16 precision yielded the same top-1 performance on ImageNet-21k as full FP32—while cutting training time nearly in half due to increased accelerator throughput [63]. In natural language processing, the results are more nuanced. While encoder-decoder models such as T5 and BART show reasonable resilience to precision loss, autoregressive decoders like GPT-style transformers are considerably more sensitive to quantization [64]. This is primarily due to the accumulation of rounding errors in long-context attention mechanisms and residual pathways [65]. However, several studies have shown that with careful tuning, mixed-precision training is still viable even at massive scales [66]. For example, OpenAI has reported the successful use of FP16 and TF32 arithmetic in the pretraining of GPT-3 using a dense transformer with 175 billion parameters. Although the exact training infrastructure was not disclosed in detail, it is widely accepted that memory and throughput savings from reduced-precision formats were crucial in making such scale economically feasible [67]. More recent work on BLOOM and LLaMA models by Hugging Face and Meta AI respectively has corroborated these findings, with bfloat16-based training pipelines demonstrating strong convergence properties and stable long-range generation [68]. Moreover, the emergence of precision-aware libraries such as DeepSpeed, Megatron-LM, and FairScale has enabled robust handling of numerical instabilities, including fine-grained control over attention scores, normalization layers, and gradient clipping—all critical components for low-precision success in large language model (LLM) settings. In the domain of extreme quantization, researchers have begun exploring the feasibility of training at 8-bit and even 4-bit precision throughout the entire training loop. One milestone achievement in this regard is the work on 8-bit optimizers by Dettmers et al., who introduced QAdam and QSGD—quantized versions of the Adam and SGD optimizers that store and update momentum buffers in 8-bit fixed-point formats. Their experiments demonstrated that models such as BERT-base and RoBERTa could be fine-tuned on GLUE benchmarks using only 8-bit weights, activations, and optimizers, with negligible degradation in accuracy [69]. These results highlight that under certain conditions, 8-bit end-to-end training is not only possible but effective—particularly in the fine-tuning regime where the pre-trained weights already encode much of the necessary inductive bias [70]. Going further, recent studies have shown preliminary results for 4-bit training using quantization-aware training, layer-wise adaptive scaling, and stochastic rounding. Though still nascent, this research suggests that the future of training may not be bounded by FP16 or bfloat16, but could extend to truly minimal representations, unlocking new levels of efficiency in data center and edge computing environments. Benchmarks from large-scale industrial labs provide additional insight into the real-world implications of low-precision training. For instance, Microsoft's DeepSpeed team has released extensive benchmarks showing that mixed-precision training with FP16 and ZeRO-offloading can scale GPT-style models to over 1 trillion parameters using hundreds of GPUs with significant reductions in memory consumption and wall-clock time. Similarly, NVIDIA's NeMo Megatron has demonstrated that using FP8 arithmetic on Hopper GPUs enables training of 22B-parameter transformer models with a 2× speedup over FP16 baselines, while maintaining BLEU and perplexity scores on par with full-precision training. These benchmarks are particularly important because they include end-to-end pipeline effects, such as distributed training overhead, gradient synchronization, and checkpointing—all of which can be bottlenecks in real-world settings and must be included when evaluating the net benefit of low-precision methods. Despite these successes, it is important to recognize that the application of low-precision training is not without failure modes. Several case studies have reported convergence collapse or accuracy degradation when low-bit quantization is applied naively. In some tasks, such as reinforcement learning or adversarial training, the instability introduced by quantization noise can compound existing volatility in optimization dynamics, leading to brittle or unstable training. Furthermore, models that rely heavily on high dynamic range computations—such as wavelet-based audio transformers or scientific machine learning models—may be inherently less amenable to low-precision computation without domain-specific modifications. Finally, many published benchmarks rely on extensive engineering effort to succeed, including gradient clipping, dynamic loss scaling, layer-wise precision tuning, and batch size adjustments—efforts that may not generalize or be available to every user or deployment context [71]. In conclusion, the case studies and benchmarks to date paint a compelling picture: low-precision training is not only a promising direction for efficiency, it is increasingly becoming the de facto standard for training large-scale foundation models. However, its adoption requires deliberate engineering, a deep understanding of model-specific precision sensitivities, and careful benchmarking. As hardware continues to evolve and software toolchains mature, we can expect even more aggressive forms of quantization to become viable at scale. Nevertheless, robust empirical evaluation across diverse tasks, architectures, and training regimes remains essential to fully realizing the potential of low-precision training in practice [72].

8. Theoretical Insights into Low-Precision Training

While much of the progress in low-precision training has been empirical, a growing body of theoretical work seeks to understand the mathematical underpinnings of quantized optimization. These analyses aim to answer fundamental questions: Under what conditions does low-precision training converge? How does quantization affect generalization? What is the optimal balance between precision, noise, and step size in gradient-based methods? Despite significant advances in algorithmic design and hardware implementation, theoretical guarantees for quantized learning remain sparse, often specific to particular regimes or assumptions. Nonetheless, the insights gleaned from these studies are invaluable for designing robust quantization strategies that scale to large foundation models without sacrificing convergence guarantees [73]. At the core of the theoretical challenge is the fact that quantization introduces a deterministic or stochastic distortion to the optimization trajectory. Let us consider the standard stochastic gradient descent (SGD) update under quantization:

$$\theta_{t+1} = \mathcal{Q}_p(\theta_t - \eta \cdot \mathcal{Q}_p(g_t)),$$

where $\mathcal{Q}_p$ is a quantization operator mapping floating-point values to a finite discrete set defined by p bits, and $g_t = \nabla \mathcal{L}_{\mathcal{B}}(\theta_t)$ is the mini-batch gradient at iteration t. The two applications of $\mathcal{Q}_p$—one to the gradient and one to the weight update—each inject noise into the optimization process. The resulting update rule can be viewed as a perturbed version of full-precision SGD [74]. A typical assumption in the theoretical literature is that $\mathcal{Q}_p$ satisfies the property of unbiasedness in expectation, i.e., $\mathbb{E}[\mathcal{Q}_p(x)] = x$, and that the variance of the quantization error is bounded: $\mathrm{Var}[\mathcal{Q}_p(x) - x] \leq \sigma_q^2$ [75]. Under these conditions, convergence results similar to standard SGD can be derived, albeit with additional error terms that depend on σ_q^2, the step size η, and the curvature of the loss landscape. However, these assumptions often fail in practical implementations. Many quantization schemes used in hardware or frameworks such as QAT, QSGD, or DoReFa-Net do not yield strictly unbiased estimators, and their noise is highly non-uniform across magnitudes and layers [76]. Moreover, in the presence of sharp minima, saddle points, or highly anisotropic loss surfaces—features common in transformer-based models—the impact of quantization noise can be amplified [77]. Some theoretical analyses attempt to sidestep these difficulties by modeling quantization noise as Gaussian perturbations added to gradients, transforming the SGD update into a discretized stochastic differential equation (SDE). In this framework, low-precision SGD is treated as a stochastic process:

$$d\theta_t = -\nabla \mathcal{L}(\theta_t)dt + \sqrt{2\sigma_q^2} dW_t,$$

where W_t is a Wiener process. This formulation allows researchers to analyze convergence in expectation or in distribution, depending on whether the goal is optimization (e.g., minimizing training loss) or generalization (e.g., finding flat minima) [78]. Results from stochastic optimization theory suggest that noise of the right magnitude can actually aid generalization by preventing overfitting to sharp minima and encouraging exploration of wide valleys in the loss surface. Thus, under certain conditions, quantization-induced noise may be beneficial—a hypothesis that is supported by some empirical findings [79]. Another theoretical line of inquiry focuses on the Lipschitz continuity and smoothness properties of the loss function under quantized updates. In full precision, convergence of SGD often assumes that the loss $\mathcal{L}(\theta)$ is L-smooth, i.e., it satisfies:

$$\|\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta')\| \leq L\|\theta - \theta'\|$$

[51]. However, quantized updates introduce discretization that violates this continuity in a strict sense. To address this, some works define relaxed notions of "quantized smoothness" or use piecewise approximations of the loss function. Others analyze quantized training through the lens of compressed sensing or sketching theory, where updates are projected into a lower-dimensional subspace and then reconstructed via decoding schemes. These methods yield convergence bounds that depend on sparsity assumptions, gradient norms, and quantization resolution, typically expressed in terms of quantization bits p and layer-wise scale parameters [80]. The optimization dynamics in low-precision regimes also interact strongly with learning rate schedules. For example, large step sizes in early training can help mitigate quantization noise by overwhelming small errors, but they can also cause divergence if quantization scales are poorly initialized [81]. Conversely, small step sizes near convergence may be rendered ineffective if quantization noise dominates the update signal [82]. This trade-off suggests a need for precision-aware learning rate schedules or adaptive mechanisms that co-tune precision and step size. Some recent proposals incorporate learning rate annealing with progressive precision reduction—starting with higher precision and gradually introducing more aggressive quantization [83]. This is loosely analogous to simulated annealing in optimization theory, where temperature is reduced to refine the solution. The theoretical analysis of such schemes remains an open problem but shows promise in bridging the gap between stability and efficiency. Furthermore, generalization under quantization remains an elusive yet critical question. While overparameterized neural networks have been shown to generalize well despite interpolation of training data, it is unclear whether this robustness extends to models trained in aggressively quantized regimes. Some theoretical models posit that quantization acts as a regularizer, enforcing implicit sparsity or weight decay. Others argue that quantization may destroy fine-grained features essential for downstream transfer, especially in low-resource or domain-shifted settings [84]. PAC-Bayesian analyses and information-theoretic bounds have been adapted to account for quantization noise, but these results often depend on strong assumptions about gradient distributions, prior knowledge, or compression schemes. As a result, we still lack a unified theory that explains when and why low-precision training generalizes well across architectures and tasks. In summary, the theoretical landscape of low-precision training is rich with insights yet full of unanswered questions. While foundational convergence guarantees have been established under idealized assumptions, much remains to be understood about the interaction between precision, noise, curvature, and generalization [85]. Bridging this gap will require a synthesis of stochastic optimization, numerical analysis, information theory, and deep learning theory. Ultimately, a mature theoretical framework will not only explain existing empirical successes but will also guide the development of new algorithms, architectures, and precision formats that are fundamentally compatible with efficient learning at scale.

9. Open Challenges and Future Directions

Despite the significant advances in low-precision training of large foundation models, numerous unresolved challenges remain—both practical and theoretical—that must be addressed to realize the full potential of this paradigm [86]. As we move toward ever-larger models, increasingly heterogeneous hardware environments, and expanding application domains, the design space for quantized training becomes more complex, and the associated risks and trade-offs become more pronounced. In this section, we elaborate on key open problems in the field, highlight emerging research directions, and outline a roadmap for the next generation of scalable, efficient, and reliable low-precision learning systems [87]. One of the most pressing challenges lies in the **robustness and generality** of low-precision techniques across model architectures and training regimes. While transformer-based models dominate much of the landscape in NLP, vision, and multimodal domains, other architectures—such as diffusion models, graph neural networks (GNNs), and neural ODEs—exhibit fundamentally different numerical behavior that may not tolerate aggressive quantization. For instance, the stochasticity and iterative refinement in diffusion models make them particularly sensitive to perturbations in their denoising steps, while GNNs often rely on sparse matrix multiplications and high dynamic range aggregations, which are not easily quantized without custom schemes. In these settings, standard quantization-aware training or mixed-precision heuristics may fail catastrophically. Moreover, foundational models are increasingly trained in multi-task, multi-modal, or continual learning setups, where shifts in data distribution or task structure can interact unpredictably with quantization noise, leading to instabilities or performance degradation. As such, designing precision-robust architectures and training algorithms that gracefully adapt to structural diversity and shifting objectives remains an open research frontier [88]. Equally critical is the problem of **precision scheduling and adaptivity**. Most current systems fix a global precision configuration prior to training—e.g., using FP16 or bfloat16 throughout, or maintaining a set of quantization parameters per layer. However, this static approach ignores the temporal and spatial heterogeneity of training dynamics. In reality, different layers, tensors, and time steps require different degrees of numerical fidelity [89]. Early layers may be more robust to noise due to their general-purpose features, while deeper layers (e.g., attention mechanisms and MLP heads) may demand higher precision to encode complex hierarchical patterns. Similarly, early training phases may tolerate lower precision due to the scale of gradients and the presence of stochastic exploration, whereas later phases may require finer updates as optimization converges. There is growing evidence that precision should be treated as a dynamic hyperparameter—co-evolving with learning rate, batch size, and other training signals. Proposed solutions include layer-wise precision tuning via reinforcement learning, entropy-based precision scaling, and gradient-aware quantization [90]. However, these techniques remain expensive, under-tested at scale, and sensitive to hyperparameter initialization. A unified framework for precision scheduling—one that incorporates feedback from optimization metrics and statistical geometry—could be transformative in making low-precision training robust and widely deployable [91]. Another unresolved issue concerns **optimization landscapes and convergence under extreme quantization**. As we push precision to 8-bit, 4-bit, or even binary regimes, the underlying loss surfaces become non-differentiable and piecewise-discontinuous, which violates the assumptions of classical gradient-based methods. Traditional notions of smoothness, Lipschitz continuity, and curvature are no longer applicable, making theoretical analysis and algorithm design significantly harder [92]. While surrogate gradient methods and stochastic smoothing techniques have been proposed, they often come at the cost of additional noise or computational overhead [93]. Furthermore, low-precision training is frequently accompanied by non-convexities arising from auxiliary components such as layer normalization, activation functions, and residual connections. This leads to complex interactions between quantization, regularization, and network topology that are poorly understood. The development of new optimization frameworks—perhaps inspired by control theory, game theory, or robust statistics—may be necessary to navigate these irregular landscapes and offer convergence guarantees that hold under minimal precision. From a systems perspective, **hardware-software co-design** is both a challenge and an opportunity. Specialized hardware such as NVIDIA's Tensor Cores, Google's TPU systolic arrays, and AMD's ROCm accelerators offer native support for low-precision arithmetic (e.g., FP16, bfloat16, and even FP8). However, actual gains depend critically on compiler behavior, memory alignment, instruction throughput, and communication overhead in distributed settings [94]. Training large models on such platforms often involves complex graph transformations, mixed-precision kernels, and precision-specific optimization paths that are tightly coupled to hardware architecture [95]. Unfortunately, this tight coupling introduces brittleness: changes in precision policy may require rewrites of entire training loops, complicating reproducibility and portability. Moreover, emerging hardware platforms—such as AI edge chips, neuromorphic processors, and optical computing systems—bring with them entirely new numerical constraints and energy models that are not well supported by current toolchains. A promising direction here is the development of **hardware-agnostic abstraction layers** and compiler optimizations that treat precision as a first-class citizen, enabling seamless experimentation, tuning, and deployment across heterogeneous environments [96]. The issue of **fairness, calibration, and interpretability** in low-precision models is also under-explored. As foundation models are increasingly deployed in high-stakes settings—ranging from medical diagnostics to legal analysis—ensuring that their predictions are stable, explainable, and free from spurious correlations becomes paramount [97]. However, low-precision arithmetic may introduce subtle biases in the representation of inputs and outputs, especially when dealing with categorical embeddings, probabilistic classifiers, or multi-modal fusion [98]. Quantization noise can disproportionately affect underrepresented features, leading to degraded performance for specific groups or tasks. Furthermore, post-hoc explainability methods such as SHAP or saliency maps may become unreliable under low precision, as small changes in internal activations are magnified by quantized gradients [99]. Future work must address these concerns by integrating fairness constraints, interpretable regularizers, and robust calibration metrics into the training process itself—ideally in a precision-aware manner that compensates for the distortions introduced by quantization [100]. Finally, a broader vision for the future of low-precision training must address the **interplay between model scaling and energy efficiency** [101]. While quantization is often motivated by speed or memory constraints, its most profound impact may lie in its potential to decouple model size from environmental cost. Training a trillion-parameter model at FP16 requires orders of magnitude more energy than training a comparable model at 8-bit or 4-bit precision. As AI systems become a major component of global energy consumption, reducing precision without sacrificing capability could offer an unprecedented opportunity for sustainable AI [102]. Realizing this potential will require not just algorithmic ingenuity, but also industry-wide benchmarks, regulatory frameworks, and carbon-aware infrastructure [103]. It may also motivate a rethinking of model evaluation itself: should we value performance per watt as highly as raw accuracy [104]? Can we design objective functions that internalize energy and resource costs? These questions remain largely open, but they are crucial for the responsible scaling of AI in the decades to come. In conclusion, low-precision training represents one of the most promising frontiers in scalable machine learning, offering a compelling path toward faster, cheaper, and greener AI. Yet, to fulfill this promise, the community must tackle a wide array of technical, theoretical, and ethical challenges. Progress will depend on deeper collaboration between algorithm designers, hardware engineers, theoreticians, and practitioners [105]. By confronting these challenges head-on—and by grounding innovation in rigorous empirical and theoretical work—we can build foundation models that are not only large and powerful, but also efficient, robust, and equitable.

10. Conclusions

The rapid expansion of large foundation models has catalyzed a new era of machine learning, one in which scale is both the driving force and the primary bottleneck. While the capabilities of these models continue to surpass expectations—powering breakthroughs in natural language processing, vision, code generation, biology, and multimodal reasoning—the cost of training, deploying, and fine-tuning them has spiraled into a domain accessible only to the most well-funded institutions. This imbalance poses critical questions about the future trajectory of AI research and deployment: Can we democratize large-scale learning without compromising performance? Can we tame the computational hunger of trillion-parameter networks without redesigning the entire training pipeline? Can precision itself become a tunable axis of efficiency, robustness, and generalization? Low-precision training offers an affirmative direction—a promising convergence of numerical optimization, systems design, and algorithmic ingenuity that enables us to push the frontier of model performance while remaining sensitive to constraints of memory, speed, and energy.

Through this survey, we have charted the conceptual, empirical, and theoretical landscape of low-precision training as it applies to large foundation models. We began by motivating the need for reduced-precision computation, tracing the evolution of numerical formats from FP32 to FP16, bfloat16, INT8, FP8, and even sub-8-bit regimes. We highlighted the computational benefits of such formats across training and inference, while also emphasizing the complexity introduced by quantization errors, representational limitations, and numerical instability. We examined various algorithmic strategies—quantization-aware training, mixed-precision optimization, stochastic quantizers, precision calibration, and layer-wise scaling—all of which aim to maintain fidelity while reducing cost. In parallel, we presented theoretical insights into how quantized SGD behaves under stochastic noise, how convergence and generalization are affected by discretization, and how the loss landscape is reshaped by low-bit updates. We also explored practical considerations: implementation on modern accelerators, precision-aware learning rate schedules, gradient clipping under quantization, and post-training quantization for deployment scenarios. Our discussion included case studies of quantized training for vision transformers, LLMs, and encoder-decoder architectures, offering empirical context for the abstract formulations.

Yet, despite the encouraging results and active research in this space, we emphasized that low-precision training is far from a solved problem. Open challenges persist in adapting precision strategies across diverse architectures, developing dynamic precision schedulers that evolve during training, and extending low-bit regimes to tasks that involve long-range dependencies, probabilistic inference, or continual learning. Theoretical guarantees remain sparse and often fail to account for the realities of transformer-based optimization, while hardware-software co-design requires greater abstraction and modularity to enable flexible experimentation. We discussed the ethical implications of quantization, especially as it affects model bias, interpretability, and fairness, noting that the interaction between quantization noise and data heterogeneity may exacerbate model instability in underrepresented groups or domains. Finally, we reflected on the energy landscape of AI, arguing that low-precision training—when paired with scalable architecture and software innovation—may be the most promising route toward sustainable and equitable AI infrastructure in the long term.

In essence, the journey toward efficient, low-precision training is not just about compressing models or accelerating matrix operations—it is about redefining the foundation of scalable learning itself. It challenges us to reconsider the relationships between information, representation, and computation, and to build systems that are not merely powerful, but also efficient, transparent, and accessible. As research in this domain continues to evolve—bridging gaps between hardware engineering, algorithmic theory, and application-driven demands—we believe that low-precision training will not only remain relevant but will emerge as a defining element in the design of the next generation of foundation models. The future of AI will not be built solely on scale, but on the principled and precise orchestration of limited computational resources to achieve maximal learning utility. Precision, in this sense, is not a constraint, but a design opportunity. And it is in this opportunity that the field of low-precision training finds its most exciting and enduring potential.

References

1. Das, D.; Mellempudi, N.; Mudigere, D.; Kalamkar, D.D.; Avancha, S.; Banerjee, K.; Sridharan, S.; Vaidyanathan, K.; Kaul, B.; Georganas, E.; et al. Mixed Precision Training of Convolutional Neural Networks using Integer Operations. In Proceedings of the International Conference on Learning Representations, 2018.
2. Kalamkar, D.; Mudigere, D.; Mellempudi, N.; Das, D.; Banerjee, K.; Avancha, S.; Vooturi, D.T.; Jammalamadaka, N.; Huang, J.; Yuen, H.; et al. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322* **2019**.
3. Panferov, A.; Chen, J.; Tabesh, S.; Castro, R.L.; Nikdan, M.; Alistarh, D. QuEST: Stable Training of LLMs with 1-Bit Weights and Activations. *arXiv preprint arXiv:2502.05003* **2025**.
4. Chen, C.; Shen, L.; Huang, H.; Liu, W. Quantized Adam with Error Feedback. *ACM Transactions on Intelligent Systems and Technology* **2021**, *12*, 56:1–56:26.
5. Zhao, K.; Tabaru, T.; Kobayashi, K.; Honda, T.; Yamazaki, M.; Tsuruoka, Y. Direct Quantized Training of Language Models with Stochastic Rounding. *arXiv preprint arXiv:2412.04787* **2024**.

6. Chen, J.; Zheng, L.; Yao, Z.; Wang, D.; Stoica, I.; Mahoney, M.W.; Gonzalez, J. ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training. In Proceedings of the International Conference on Machine Learning, 2021, pp. 1803–1813.
7. Ma, Y.; Yu, D.; Wu, T.; Wang, H. PaddlePaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing* **2019**, *1*, 105–115.
8. He, X.; Sun, J.; Chen, H.; Li, D. Campo: Cost-Aware Performance Optimization for Mixed-Precision Neural Network Training. In Proceedings of the USENIX Annual Technical Conference, 2022, pp. 505–518.
9. Micikevicius, P.; Stosic, D.; Burgess, N.; Cornea, M.; Dubey, P.; Grisenthwaite, R.; Ha, S.; Heinecke, A.; Judd, P.; Kamalu, J.; et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433* **2022**.
10. Zhang, Z.; Jaiswal, A.; Yin, L.; Liu, S.; Zhao, J.; Tian, Y.; Wang, Z. Q-galore: Quantized galore with int4 projection and layer-adaptive low-rank gradients. *arXiv preprint arXiv:2407.08296* **2024**.
11. Xie, X.; Lin, Z.; Toh, K.C.; Zhou, P. Loco: Low-bit communication adaptor for large-scale model training. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2025**.
12. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the International Conference on Machine Learning, 2015, pp. 448–456.
13. Faghri, F.; Tabrizian, I.; Markov, I.; Alistarh, D.; Roy, D.M.; Ramezani-Kebrya, A. Adaptive Gradient Quantization for Data-Parallel SGD. In Proceedings of the Advances in Neural Information Processing Systems, 2020.
14. Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; Sutskever, I. Zero-Shot Text-to-Image Generation. In Proceedings of the International Conference on Machine Learning, 2021, pp. 8821–8831.
15. Fu, Y.; You, H.; Zhao, Y.; Wang, Y.; Li, C.; Gopalakrishnan, K.; Wang, Z.; Lin, Y. FracTrain: Fractionally Squeezing Bit Savings Both Temporally and Spatially for Efficient DNN Training. In Proceedings of the Advances in Neural Information Processing Systems, 2020.
16. Karimireddy, S.P.; Rebjock, Q.; Stich, S.U.; Jaggi, M. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In Proceedings of the International Conference on Machine Learning, 2019, pp. 3252–3261.
17. Modoranu, I.; Safaryan, M.; Malinovsky, G.; Kurtic, E.; Robert, T.; Richtárik, P.; Alistarh, D. MicroAdam: Accurate Adaptive Optimization with Low Space Overhead and Provable Convergence. In Proceedings of the Advances in Neural Information Processing Systems, 2024.
18. Li, J.; Ding, K.; Toh, K.C.; Zhou, P. Memory-Efficient 4-bit Preconditioned Stochastic Optimization. *arXiv preprint arXiv:2412.10663* **2024**.
19. Fishman, M.; Chmiel, B.; Banner, R.; Soudry, D. Scaling FP8 training to trillion-token LLMs. *arXiv preprint arXiv:2409.12517* **2024**.
20. Shazeer, N.; Stern, M. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In Proceedings of the International Conference on Machine Learning, 2018, pp. 4603–4611.
21. Gao, C.; Chen, J.; Zhao, K.; Wang, J.; Jing, L. 1-Bit FQT: Pushing the Limit of Fully Quantized Training to 1-bit. *arXiv preprint arXiv:2408.14267* **2024**.
22. Dettmers, T.; Lewis, M.; Belkada, Y.; Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339* **2022**.
23. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342* **2018**.
24. Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160* **2016**.
25. Liu, Z.; Zhou, K.; Yang, F.; Li, L.; Chen, R.; Hu, X. EXACT: Scalable Graph Neural Networks Training via Extreme Activation Compression. In Proceedings of the International Conference on Learning Representations, 2022.
26. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In Proceedings of the European Conference on Computer Vision, 2016, pp. 525–542.
27. Chitsaz, K.; Fournier, Q.; Mordido, G.; Chandar, S. Exploring Quantization for Efficient Pre-Training of Transformer Language Models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2024, pp. 13473–13487.
28. Harma, S.B.; Chakraborty, A.; Sperry, N.; Falsafi, B.; Jaggi, M.; Oh, Y. Accuracy Booster: Enabling 4-bit Fixed-point Arithmetic for DNN Training. *arXiv preprint arXiv:2211.10737* **2022**.

29. Yang, Y.; Deng, L.; Wu, S.; Yan, T.; Xie, Y.; Li, G. Training high-performance and large-scale deep neural networks with full 8-bit integers. *Neural Networks* **2020**, *125*, 70–82.
30. Chen, X.; Liang, C.; Huang, D.; Real, E.; Wang, K.; Pham, H.; Dong, X.; Luong, T.; Hsieh, C.; Lu, Y.; et al. Symbolic Discovery of Optimization Algorithms. In Proceedings of the Advances in Neural Information Processing Systems, 2023.
31. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLoRA: Efficient Finetuning of Quantized LLMs. In Proceedings of the Advances in Neural Information Processing Systems, 2023.
32. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; Bengio, Y.; LeCun, Y., Eds., 2015.
33. Seide, F.; Fu, H.; Droppo, J.; Li, G.; Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. In Proceedings of the Annual Conference of the International Speech Communication Association, 2014, pp. 1058–1062.
34. Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* **2024**.
35. Dettmers, T. 8-bit approximations for parallelism in deep learning. *arXiv preprint arXiv:1511.04561* **2015**.
36. Han, R.; Demmel, J.; You, Y. Auto-precision scaling for distributed deep learning. In Proceedings of the High Performance Computing: 36th International Conference, ISC High Performance 2021, Virtual Event, June 24–July 2, 2021, Proceedings 36. Springer, 2021, pp. 79–97.
37. Chen, Y.; Xi, H.; Zhu, J.; Chen, J. Oscillation-Reduced MXFP4 Training for Vision Transformers. *arXiv preprint arXiv:2502.20853* **2025**.
38. Pang, J.; Cai, T. Stabilizing Quantization-Aware Training by Implicit-Regularization on Hessian Matrix. *arXiv preprint arXiv:2503.11159* **2025**.
39. Blake, C.; Orr, D.; Luschi, C. Unit scaling: Out-of-the-box low-precision training. In Proceedings of the International Conference on Machine Learning. PMLR, 2023, pp. 2548–2576.
40. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations; Bengio, Y.; LeCun, Y., Eds., 2015.
41. Lee, J.; Bae, J.; Kim, B.; Kwon, S.J.; Lee, D. To fp8 and back again: Quantifying the effects of reducing precision on llm training stability. *arXiv preprint arXiv:2405.18710* **2024**.
42. Yu, Y.; Wu, J.; Huang, L. Double Quantization for Communication-Efficient Distributed Optimization. In Proceedings of the Advances in Neural Information Processing Systems, 2019, pp. 4440–4451.
43. Ding, L.; Fei, W.; Huang, Y.; Ding, S.; Dai, W.; Li, C.; Zou, J.; Xiong, H. AMPA: Adaptive Mixed Precision Allocation for Low-Bit Integer Training. In Proceedings of the International Conference on Machine Learning, 2024.
44. Nielsen, J.; Schneider-Kamp, P.; Galke, L. Continual Quantization-Aware Pre-Training: When to transition from 16-bit to 1.58-bit pre-training for BitNet language models? *arXiv preprint arXiv:2502.11895* **2025**.
45. Zhang, P.; Wei, J.; Zhang, J.; Zhu, J.; Chen, J. Accurate INT8 Training Through Dynamic Block-Level Fallback. *arXiv preprint arXiv:2503.08040* **2025**.
46. Hinton, G.E.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv: 1503.02531* **2015**.
47. Banner, R.; Hubara, I.; Hoffer, E.; Soudry, D. Scalable methods for 8-bit training of neural networks. In Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 5151–5159.
48. Dettmers, T.; Lewis, M.; Shleifer, S.; Zettlemoyer, L. 8-bit Optimizers via Block-wise Quantization. *arXiv preprint arXiv:2110.02861* **2021**.
49. Sun, X.; Choi, J.; Chen, C.; Wang, N.; Venkataramani, S.; Srinivasan, V.; Cui, X.; Zhang, W.; Gopalakrishnan, K. Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, 2019, pp. 4901–4910.
50. Zhang, X.; Liu, S.; Zhang, R.; Liu, C.; Huang, D.; Zhou, S.; Guo, J.; Guo, Q.; Du, Z.; Zhi, T.; et al. Fixed-Point Back-Propagation Training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2327–2335.
51. Zniyed, Y.; Nguyen, T.P.; et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems* **2024**, *36*, 4358–4370.
52. Zhao, Y.; Lin, C.Y.; Zhu, K.; Ye, Z.; Chen, L.; Zheng, S.; Ceze, L.; Krishnamurthy, A.; Chen, T.; Kasikci, B. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems* **2024**, pp. 196–209.

53. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.
54. Li, Z.; Sa, C.D. Dimension-Free Bounds for Low-Precision Training. In Proceedings of the Advances in Neural Information Processing Systems, 2019, pp. 11728–11738.
55. Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; Vojnovic, M. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 1709–1720.
56. Sylvester, J.J. LX. Thoughts on inverse orthogonal matrices, simultaneous signsuccessions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **1867**, *34*, 461–475.
57. Eliassen, S.; Selvan, R. Activation Compression of Graph Neural Networks Using Block-Wise Quantization with Improved Variance Minimization. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2024, pp. 7430–7434.
58. Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.M.; Wang, W.C.; Xiao, G.; Dang, X.; Gan, C.; Han, S. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems* **2024**, pp. 87–100.
59. Zhu, F.; Gong, R.; Yu, F.; Liu, X.; Wang, Y.; Li, Z.; Yang, X.; Yan, J. Towards Unified INT8 Training for Convolutional Neural Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1966–1976.
60. Tang, H.; Yu, C.; Lian, X.; Zhang, T.; Liu, J. DoubleSqueeze: Parallel Stochastic Gradient Descent with Double-pass Error-Compensated Compression. In Proceedings of the International Conference on Machine Learning, 2019, pp. 6155–6165.
61. Balança, P.; Hosegood, S.; Luschi, C.; Fitzgibbon, A. Scalify: scale propagation for efficient low-precision LLM training. *arXiv preprint arXiv:2407.17353* **2024**.
62. Desrentes, O.; de Dinechin, B.D.; Le Maire, J. Exact dot product accumulate operators for 8-bit floating-point deep learning. In Proceedings of the 2023 26th Euromicro Conference on Digital System Design (DSD). IEEE, 2023, pp. 642–649.
63. Ali, S.B.; Filip, S.; Sentieys, O. A Stochastic Rounding-Enabled Low-Precision Floating-Point MAC for DNN Training. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, 2024, pp. 1–6.
64. Guo, W.; Liu, D.; Xie, W.; Li, Y.; Ning, X.; Meng, Z.; Zeng, S.; Lei, J.; Fang, Z.; Wang, Y. Towards Accurate and Efficient Sub-8-Bit Integer Training. *arXiv preprint arXiv:2411.10948* **2024**.
65. Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; Narayanan, P. Deep Learning with Limited Numerical Precision. In Proceedings of the International Conference on Machine Learning; Bach, F.R.; Blei, D.M., Eds., 2015, pp. 1737–1746.
66. Fei, W.; Dai, W.; Zhang, L.; Zhang, L.; Li, C.; Zou, J.; Xiong, H. Latent Weight Quantization for Integerized Training of Deep Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2025**, *47*, 2816–2832.
67. Yang, J.; Deng, L.; Yang, Y.; Xie, Y.; Li, G. Training and inference for integer-based semantic segmentation network. *Neurocomputing* **2021**, *454*, 101–112.
68. Dai, D.; Zhang, Y.; Zhang, J.; Hu, Z.; Cai, Y.; Sun, Q.; Zhang, Z. Trainable Fixed-Point Quantization for Deep Learning Acceleration on FPGAs. *arXiv preprint arXiv:2401.17544* **2024**.
69. Xia, L.; Anthonissen, M.; Hochstenbach, M.; Koren, B. A simple and efficient stochastic rounding method for training neural networks in low precision. *arXiv preprint arXiv:2103.13445* **2021**.
70. Courbariaux, M.; Bengio, Y.; David, J. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In Proceedings of the Advances in Neural Information Processing Systems, 2015, pp. 3123–3131.
71. Sun, X.; Wang, N.; Chen, C.; Ni, J.; Agrawal, A.; Cui, X.; Venkataramani, S.; Maghraoui, K.E.; Srinivasan, V.; Gopalakrishnan, K. Ultra-Low Precision 4-bit Training of Deep Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, 2020.
72. Kim, S.; Park, E. HLQ: Fast and Efficient Backpropagation via Hadamard Low-rank Quantization. *arXiv preprint arXiv:2406.15102* **2024**.
73. Zhao, K.; Huang, S.; Pan, P.; Li, Y.; Zhang, Y.; Gu, Z.; Xu, Y. Distribution Adaptive INT8 Quantization for Training CNNs. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 3483–3491.

74. Novikov, G.S.; Bershatsky, D.; Gusak, J.; Shonenkov, A.; Dimitrov, D.V.; Oseledets, I.V. Few-bit Backward: Quantized Gradients of Activation Functions for Memory Footprint Reduction. In Proceedings of the International Conference on Machine Learning, 2023, pp. 26363–26381.
75. Köster, U.; Webb, T.; Wang, X.; Nassar, M.; Bansal, A.K.; Constable, W.; Elibol, O.; Hall, S.; Hornof, L.; Khosrowshahi, A.; et al. Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 1742–1752.
76. Park, H.; Lee, J.H.; Oh, Y.; Ha, S.; Lee, S. Training deep neural network in limited precision. *arXiv preprint arXiv:1810.05486* **2018**.
77. Yang, Y.; Gao, J.; Hu, W. RaanA: A Fast, Flexible, and Data-Efficient Post-Training Quantization Algorithm. *arXiv preprint arXiv:2504.03717* **2025**.
78. Courbariaux, M.; Bengio, Y.; David, J.P. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024* **2014**.
79. Chen, X.; Hu, X.; Zhou, H.; Xu, N. FxpNet: Training a deep convolutional neural network in fixed-point representation. In Proceedings of the International Joint Conference on Neural Networks, 2017, pp. 2494–2501.
80. Du, D.; Zhang, Y.; Cao, S.; Guo, J.; Cao, T.; Chu, X.; Xu, N. BitDistiller: Unleashing the Potential of Sub-4-Bit LLMs via Self-Distillation. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2024, pp. 102–116.
81. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
82. Wang, G.; Qin, H.; Jacobs, S.A.; Holmes, C.; Rajbhandari, S.; Ruwase, O.; Yan, F.; Yang, L.; He, Y. Zero++: Extremely efficient collective communication for giant model training. *arXiv preprint arXiv:2306.10209* **2023**.
83. Drumond, M.; Lin, T.; Jaggi, M.; Falsafi, B. Training DNNs with Hybrid Block Floating Point. In Proceedings of the Advances in Neural Information Processing Systems, 2018, pp. 451–461.
84. De Dinechin, F.; Forget, L.; Muller, J.M.; Uguen, Y. Posits: the good, the bad and the ugly. In Proceedings of the Proceedings of the Conference for Next Generation Arithmetic 2019, 2019, pp. 1–10.
85. Li, S.; Liu, H.; Bian, Z.; Fang, J.; Huang, H.; Liu, Y.; Wang, B.; You, Y. Colossal-ai: A unified deep learning system for large-scale parallel training. In Proceedings of the Proceedings of the 52nd International Conference on Parallel Processing, 2023, pp. 766–775.
86. Wortsman, M.; Dettmers, T.; Zettlemoyer, L.; Morcos, A.; Farhadi, A.; Schmidt, L. Stable and low-precision training for large-scale vision-language models. In Proceedings of the Advances in Neural Information Processing Systems, 2023.
87. Xi, H.; Cai, H.; Zhu, L.; Lu, Y.; Keutzer, K.; Chen, J.; Han, S. Coat: Compressing optimizer states and activation for memory-efficient fp8 training. *arXiv preprint arXiv:2410.19313* **2024**.
88. Williamson, D. Dynamically scaled fixed point arithmetic. In Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing Conference Proceedings, 1991, pp. 315–318.
89. Rajagopal, A.; Vink, D.A.; Venieris, S.I.; Bouganis, C. Multi-Precision Policy Enforced Training (MuPPET) : A Precision-Switching Strategy for Quantised Fixed-Point Training of CNNs. In Proceedings of the International Conference on Machine Learning, 2020, pp. 7943–7952.
90. Zhong, K.; Ning, X.; Dai, G.; Zhu, Z.; Zhao, T.; Zeng, S.; Wang, Y.; Yang, H. Exploring the Potential of Low-Bit Training of Convolutional Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2022**, *41*, 5421–5434.
91. Lu, J.; Fang, C.; Xu, M.; Lin, J.; Wang, Z. Evaluations on Deep Neural Networks Training Using Posit Number System. *IEEE Transactions Computers* **2021**, *70*, 174–187.
92. Wang, M.; Rasoulinezhad, S.; Leong, P.H.W.; So, H.K. NITI: Training Integer Neural Networks Using Integer-Only Arithmetic. *IEEE Transactions on Parallel and Distributed Systems* **2022**, *33*, 3249–3261.
93. Fu, Y.; Guo, H.; Li, M.; Yang, X.; Ding, Y.; Chandra, V.; Lin, Y. CPT: Efficient Deep Neural Network Training via Cyclic Precision. In Proceedings of the International Conference on Learning Representations, 2021.
94. Jia, J.; Xie, C.; Lu, H.; Wang, D.; Feng, H.; Zhang, C.; Sun, B.; Lin, H.; Zhang, Z.; Liu, X.; et al. SDP4Bit: Toward 4-bit Communication Quantization in Sharded Data Parallelism for LLM Training. In Proceedings of the Advances in Neural Information Processing Systems, 2024.
95. Chen, J.; Gai, Y.; Yao, Z.; Mahoney, M.W.; Gonzalez, J.E. A Statistical Framework for Low-bitwidth Training of Deep Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, 2020.

96. Chmiel, B.; Ben-Uri, L.; Shkolnik, M.; Hoffer, E.; Banner, R.; Soudry, D. Neural gradients are near-lognormal: improved quantized and sparse training. In Proceedings of the International Conference on Learning Representations, 2021.
97. Wang, R.; Gong, Y.; Liu, X.; Zhao, G.; Yang, Z.; Guo, B.; Zha, Z.; Cheng, P. Optimizing Large Language Model Training Using FP4 Quantization. *arXiv preprint arXiv:2501.17116* **2025**.
98. Zhao, R.; Vogel, B.; Ahmed, T.; Luk, W. Reducing Underflow in Mixed Precision Training by Gradient Scaling. In Proceedings of the International Joint Conference on Artificial Intelligence, 2020, pp. 2922–2928.
99. Ma, S.; Wang, H.; Ma, L.; Wang, L.; Wang, W.; Huang, S.; Dong, L.; Wang, R.; Xue, J.; Wei, F. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764* **2024**, *1*.
100. Li, J.; Zhang, T.; Yen, I.E.H.; Xu, D. FP8-BERT: Post-Training Quantization for Transformer. *arXiv preprint arXiv:2312.05725* **2023**.
101. Tseng, A.; Yu, T.; Park, Y. Training LLMs with MXFP4. *arXiv preprint arXiv:2502.20586* **2025**.
102. Shen, A.; Lai, Z.; Sun, T.; Li, S.; Ge, K.; Liu, W.; Li, D. Efficient deep neural network training via decreasing precision with layer capacity. *Frontiers of Computer Science* **2025**, *19*, 1910355.
103. Schiemer, M.; Schaefer, C.J.; Vap, J.P.; Horeni, M.J.; Wang, Y.E.; Ye, J.; Joshi, S. Hadamard Domain Training with Integers for Class Incremental Quantized Learning. *arXiv preprint arXiv:2310.03675* **2023**.
104. Xi, H.; Li, C.; Chen, J.; Zhu, J. Training Transformers with 4-bit Integers. In Proceedings of the Advances in Neural Information Processing Systems, 2023.
105. Shen, L.; Sun, Y.; Yu, Z.; Ding, L.; Tian, X.; Tao, D. On Efficient Training of Large-Scale Deep Learning Models. *ACM Computing Surveys* **2024**, *57*, 1–36.