Native FP4 Training for Large Language Models: Techniques, Hardware Acceleration, and Performance Analysis

1. Introduction

1.1. The Computational Challenge of Large Model Training

The field of artificial intelligence (AI) has witnessed remarkable progress, largely driven by the scaling of deep learning models, particularly Large Language models (LLMs). These models, often comprising hundreds of billions or even trillions of parameters ([1]), exhibit unprecedented capabilities in natural language understanding and generation. However, training such colossal models incurs substantial computational and memory costs, demanding vast clusters of specialized hardware and consuming significant energy and time ([9]). This escalating resource demand poses a significant barrier to research, development, and deployment, necessitating innovative approaches to enhance training efficiency. Model compression techniques, especially those leveraging low-precision numerical formats, have emerged as crucial strategies to alleviate these burdens ([12]).

1.2. FP4 Precision: The Next Frontier in Efficiency

Following the successful adoption of 16-bit floating-point formats like FP16 and BF16 ([8]) and the subsequent introduction of 8-bit floating-point (FP8) precision, particularly with hardware support in architectures like NVIDIA Hopper ([6]), the research community is now actively exploring 4-bit floating-point (FP4) precision. FP4 represents a further, aggressive step towards reducing the numerical precision used in deep learning computations. The potential advantages are compelling: a significant reduction in memory footprint for storing weights, activations, and gradients; increased computational throughput on hardware designed to support FP4 arithmetic; reduced data movement costs in distributed settings; and potentially lower energy consumption per operation ([1]). However, this aggressive reduction in bit-width introduces substantial numerical challenges. FP4 formats possess extremely limited representational capacity, leading to a restricted dynamic range and significantly reduced precision compared to higher-bit formats. This exacerbates quantization errors and increases the risk of numerical instability, such as overflow, underflow, and gradient inaccuracies during training ([12]).

1.3. Focus: Native FP4 Training

This report focuses specifically on **native FP4 training**. This term refers to training methodologies where the core computational workloads, particularly the general matrix multiplications (GEMMs) involved in the forward and potentially backward passes of neural network training, are performed using FP4 arithmetic. This is typically achieved either by training a model from scratch with simulated FP4 quantization or through Quantization-Aware Training (QAT), where a pre-trained model is fine-tuned while simulating FP4 quantization effects ([18]). This contrasts with Post-Training Quantization (PTQ), where a fully trained higher-precision model is converted to FP4 after training, primarily for inference acceleration ([18]). The feasibility and practicality of native FP4 training have been significantly boosted by the recent emergence of hardware architectures, such as NVIDIA's Blackwell ([1]), which incorporate native support for FP4 computations.

The progression towards ultra-low precision formats like FP4 exemplifies a symbiotic relationship between algorithmic innovation and hardware development. The sheer scale of modern AI models necessitates greater computational efficiency, driving the exploration of formats like FP4 ([11]). However, the inherent numerical limitations of such low precision ([13]) demand sophisticated algorithmic solutions—specialized optimizers, novel gradient handling techniques, and adaptive quantization strategies—to maintain training stability and model accuracy ([13]). Concurrently, hardware architects respond by designing accelerators (like Hopper for FP8 and Blackwell for FP4) that natively support these low-precision operations, often incorporating features assumed or targeted by the algorithmic research ([1]). This hardware enablement, in turn, accelerates further research and adoption, creating a positive feedback loop where algorithms and hardware co-evolve to push the boundaries of efficient deep learning. Progress in this domain hinges on this interplay, suggesting that future advancements will likely stem from even tighter co-design efforts.

1.4. Report Objectives and Structure

The primary objective of this report is to provide a comprehensive, technical overview of the state-of-the-art in native FP4 training for AI models, with a particular emphasis on LLMs. It delves into the fundamental characteristics of FP4 precision, details the core methodologies and specialized techniques required for stable and accurate training, analyzes the crucial role played by modern hardware accelerators like the NVIDIA Blackwell architecture, evaluates the performance impacts on inference and training, discusses the inherent trade-offs with model accuracy, and surveys recent research findings. The report aims to serve as a rigorous guide for researchers and practitioners seeking to understand and implement FP4 training.

The report is structured as follows:

- Section 2 covers the fundamentals of FP4 precision, including its numerical representation, benefits, and challenges.

- Section 3 details the core methodologies developed for native FP4 training, focusing on QAT adaptations, gradient handling, optimizers, stability techniques, and mixed-precision strategies.
- Section 4 examines the role of hardware acceleration, particularly the NVIDIA Blackwell architecture, in enabling FP4 computations.
- Section 5 analyzes the performance impact of FP4, quantifying inference acceleration, efficiency gains, and training speedup potential.
- Section 6 evaluates the trade-offs between FP4 precision and model accuracy, summarizing reported results.
- Section 7 discusses the state-of-the-art, practical implementation considerations, and outstanding challenges.
- Section 8 concludes the report with a synthesis of the key findings.

2. Fundamentals of FP4 Precision

2.1. Numerical Representation

Floating-point (FP) arithmetic provides a means to represent real numbers using a fixed number of bits, enabling the representation of values across a wide range of magnitudes ([47]). A standard FP number typically consists of three components: a sign bit (s), an exponent (e), and a significand or mantissa (m) ([47]). The value is generally interpreted as (−1)s×2exponent−bias×(1.mantissa) for normalized numbers. FP4 refers to any floating-point format that uses a total of 4 bits for these components.

Within the constraint of 4 bits, different allocations between the exponent and mantissa bits lead to distinct FP4 formats, each with unique characteristics regarding dynamic range (the span of representable magnitudes) and precision (the density of representable values). Common variants include:

- **E2M1:** This format allocates 1 bit for the sign, 2 bits for the exponent, and 1 bit for the mantissa. It offers a balance between range and precision, providing slightly better resolution than formats with no mantissa bits, particularly for values near zero ([15]). Compared to 4-bit integer (INT4) formats, E2M1 often provides better coverage of the bell-shaped or long-tailed distributions commonly observed in neural network weights and activations ([23]).
- **E3M0:** This format uses 1 sign bit, 3 exponent bits, and 0 mantissa bits. Lacking a mantissa, it essentially represents powers of 2 (and their negatives), resulting in a logarithmic-like distribution of representable values ([35]). This provides a wider dynamic range compared to E2M1 but sacrifices fractional precision between these powers of 2. Research suggests this format aligns well with the distribution of neural network gradients, which often exhibit heavy tails ([35]).

- **Radix-4 FP4 ():** Proposed by Sun et al. (2020), this format also uses 1 sign and 3 exponent bits but employs a radix (base) of 4 instead of the standard 2 for the exponent calculation ([35]). This significantly extends the dynamic range (e.g., up to 212) compared to a radix-2 E3M0 format (e.g., up to 26), which was found crucial for handling the wide range of gradient magnitudes encountered during training, especially on complex datasets like ImageNet ([43]).
- **Microscaling Formats (MXFP4):** These formats introduce a finer granularity of scaling. Instead of a single scaling factor per tensor, MX formats apply scaling factors to smaller blocks or groups of elements (e.g., groups of 32) ([3]). A common configuration, MXFP4, uses an E2M1 representation for the elements within each group, while the shared scaling factor for the group is represented using a higher-precision exponent format (e.g., 8-bit E8M0) ([27]). This approach significantly mitigates the negative impact of outlier values within a tensor, as an outlier only affects the scaling of its local group, preserving precision for other groups ([27]). MX formats, including MXFP4, are supported by the NVIDIA Blackwell architecture ([3]).

The choice between these formats often depends on the specific tensor being represented. Gradients, with their wide dynamic range, might favor E3M0 or Radix-4 formats ([35]), while weights and activations, often clustered around zero but potentially containing outliers, might benefit from E2M1 or MXFP4 ([23]). This inherent dependency on data distribution underscores that a one-size-fits-all FP4 format is unlikely to be optimal. Effective FP4 utilization often necessitates adaptive or mixed-format strategies tailored to the specific characteristics of different tensors within the model and the training process. Hardware supporting flexible FP4 representations, such as the MX formats in Blackwell, provides a significant advantage in this context ([3]).

Table 2.1 provides a comparative overview of FP4 variants alongside other common numerical formats used in deep learning.

Table 2.1: Comparison of Low-Precision Formats

Feature	**FP32 (IEEE)**	**BF16**	**FP16 (IEEE)**	**FP8 (E4M3)**	**FP8 (E5M2)**	**FP4 (E3M0)**	**FP4 (E2M1)**	**MXFP4 (E2M1/E8M0)**	**INT8**	**INT4**
Total Bits	32	16	16	8	8	4	4	4 + Shared Scale	8	4
Sign Bits	1	1	1	1	1	1	1	1	1 (signed)	1 (signed)
Exponent Bits	8	8	5	4	5	3	2	2 + 8 (shared)	N/A	N/A
Mantissa Bits	23	7	10	3	2	0	1	1	7 (signed)	3 (signed)
Dynamic Range (approx.)	Wide	Wide	Narrow	Medium	Wide	Medium	Narrow	Wide (via scale)	Fixed	Fixed
Precision (Relative)	High	Low	Medium	Low	Very Low	Very Low	Very Low	Very Low	Fixed	Fixed
Zero/Subnormal Handling	Standard	Standard	Standard	Standard	Standard	Limited	Limited	Limited	Exact Zero	Exact Zero

Key Use Case	Baseline	Training	Inference	Training	Training	Gradients?	W/A?	W/A/G (Blackwell)	Inference	Inference
Hardware Support	All	Ampere+	All	Hopper+	Hopper+	Blackwell+	Blackwell+	Blackwell+	Most Accel.	Most Accel.

(Note: Dynamic Range and Precision are relative comparisons. W/A/G = Weights/Activations/Gradients. Hardware support indicates NVIDIA architectures where native acceleration is prominent.)

2.2. Potential Benefits

The primary motivation for adopting FP4 precision lies in its potential to significantly enhance computational efficiency:

- **Reduced Memory Footprint:** Representing numerical values with only 4 bits drastically reduces the memory required for storing model parameters (weights), intermediate activations during forward and backward passes, and gradients ([22]). Compared to 16-bit formats (FP16/BF16), FP4 offers a theoretical 4x reduction in memory usage, and a 2x reduction compared to FP8 ([7]). This allows larger models to fit within limited device memory or reduces memory pressure in large-scale training systems.
- **Increased Computational Throughput:** Hardware accelerators equipped with specialized FP4 compute units, such as the Tensor Cores in NVIDIA's Blackwell GPUs, can perform significantly more operations per clock cycle compared to higher-precision units ([1]). Blackwell GPUs, for instance, claim substantial increases in theoretical FP4 PetaFLOPS performance ([2]).
- **Reduced Communication Costs:** In distributed training scenarios where gradients or parameters need to be exchanged between multiple GPUs or nodes, using FP4 reduces the amount of data that needs to be transferred across the network or interconnect, potentially alleviating communication bottlenecks ([7]).
- **Lower Power Consumption:** Performing computations and moving data with fewer bits generally requires less energy ([1]). Systems like the NVIDIA GB200 NVL72, leveraging FP4 and other optimizations like liquid cooling, claim significant improvements in energy efficiency compared to previous generations ([1]).

2.3. Intrinsic Challenges

Despite the potential benefits, the extremely low bit-width of FP4 introduces significant intrinsic challenges that must be addressed for successful training:

- **Limited Dynamic Range:** FP4 formats struggle to simultaneously represent numbers with very small and very large magnitudes ([7]). This makes them highly susceptible to overflow (values exceeding the maximum representable magnitude) and underflow (small non-zero values being flushed to zero), particularly problematic for gradients which can span several orders of magnitude during training ([16]).
- **Reduced Precision/Resolution:** With only 16 possible unique values (including signed zeros, infinities, NaNs), the gaps between representable numbers in FP4 are much larger than in higher precisions. This inherent coarseness leads to significant quantization error when converting higher-precision values to FP4 ([12]).
- **Sensitivity to Outliers:** The presence of even a few values with exceptionally large magnitudes (outliers) in a tensor can severely impact standard quantization schemes ([11]). If scaling is determined by the maximum absolute value, the presence of an outlier forces the vast majority of smaller values into a tiny portion of the FP4 range, effectively destroying their precision. This is a critical issue for both weights and activations in LLMs ([11]). Techniques like microscaling (MX) are specifically designed to combat this by localizing the impact of outliers ([3]).
- **Numerical Stability Issues:** The combination of limited range, low precision, and quantization errors can lead to severe numerical instability during training. This can manifest as exploding or vanishing gradients, loss spikes, weight oscillations around quantization boundaries, and ultimately, training divergence ([7]). Maintaining stable training dynamics requires careful algorithmic design and parameter tuning.

3. Core Methodologies for Native FP4 Training

Successfully training models natively using FP4 precision necessitates overcoming the significant numerical challenges outlined previously. This requires adopting and adapting a suite of specialized techniques that address issues across the entire training pipeline, from forward pass quantization and gradient computation to optimizer dynamics and overall stability. Simply applying FP4 precision within a standard training loop is highly likely to fail ([43]). Instead, a holistic approach integrating multiple solutions is required, recognizing that native FP4 training is fundamentally a system-level problem. Problems arising in one stage (e.g., inaccurate forward pass quantization) can exacerbate issues in subsequent stages (e.g., gradient instability).

Therefore, effective frameworks often combine techniques targeting quantization, gradient handling, optimization, and stability ([13]).

3.1. Adapting Quantization-Aware Training (QAT)

Quantization-Aware Training (QAT) is a powerful technique for mitigating accuracy loss when quantizing models, especially to very low bit-widths ([18]). Unlike PTQ, QAT simulates the effects of quantization (specifically, the rounding and clipping operations) during the model's training or fine-tuning process. This allows the model parameters to adapt to the reduced precision, often resulting in significantly better accuracy compared to applying PTQ to a model trained at higher precision ([18]).

Applying QAT for native FP4 training involves simulating FP4 quantization in the forward pass while typically using higher precision for gradient computation and weight updates. However, FP4 QAT presents unique challenges:

- **Instability:** The severe quantization introduced by FP4 can lead to unstable training dynamics and gradient mismatch issues.
- **Gradient Estimation:** The non-differentiable nature of quantization requires an estimator for backpropagation. The commonly used Straight-Through Estimator (STE), which simply passes the gradient through the quantization function, can become particularly inaccurate and unstable at the extreme quantization levels of FP4 ([20]).
- **Training Cost:** QAT inherently requires (re)training, which can be computationally expensive, potentially negating some of the efficiency gains if not managed carefully ([18]).

Several techniques have been proposed or adapted to make FP4 QAT more effective and efficient:

- **Knowledge Distillation (KD):** The FP4 model (student) is trained to mimic the outputs or internal representations of a higher-precision model (teacher) ([18]). LLM-QAT specifically uses a data-free approach where the student learns from outputs generated by the full-precision model itself ([18]).
- **Improved Gradient Estimators:** Research explores alternatives or improvements to STE, such as using differentiable approximations of the quantization function (differentiable quantization estimators) to obtain more accurate gradients for weight updates ([12]).
- **Specialized Regularization:** Techniques like saliency-aware regularization can be incorporated into the QAT loss function to prioritize the preservation of parameters deemed most important for model performance during quantization ([18]).

- **Efficient QAT Strategies:** To reduce the computational burden of full QAT, methods like block-wise fine-tuning or training only the quantization parameters (scales, zero-points) have been explored ([18]). EfficientQAT, for example, proposes a two-stage approach involving block-wise training of all parameters followed by end-to-end fine-tuning of only the quantization parameters ([45]).
- **Coreset Selection:** Utilizing smaller, representative subsets of the training data (coresets) can significantly reduce the time and resources required for QAT while potentially improving robustness by filtering noisy samples ([31]). The proposed Adaptive Coreset Selection (ACS) method dynamically selects informative samples based on metrics like error vector score and disagreement score relative to a full-precision model ([31]).

While some applications of QAT focus on optimizing models for subsequent FP4 inference ([49]), the underlying principles and techniques are directly relevant for achieving robust native FP4 training from scratch or via fine-tuning.

3.2. Gradient Computation and Handling in FP4

Gradients are the lifeblood of stochastic gradient descent (SGD)-based training methods. Accurate representation and accumulation of gradients are paramount for model convergence. However, the extreme limitations of FP4 precision pose severe challenges for gradients ([16]). Gradients often exhibit a wide dynamic range and heavy-tailed distributions, making them susceptible to both underflow (losing small but potentially important update signals) and overflow (losing information due to clipping large values) in FP4. Furthermore, the low resolution leads to significant quantization noise and potential bias in gradient estimates. Techniques to address these issues include:

- **Specialized Formats and Scaling:**
  - *Logarithmic/Radix-4 FP4:* Formats like E3M0 or Radix-4 FP4 () provide a wider dynamic range suitable for gradient distributions ([35]). However, naive use of logarithmic formats can introduce bias ([35]).
  - *Adaptive Gradient Scaling (GradScale):* Instead of a single global loss scaling factor (which is often insufficient due to varying gradient magnitudes across layers), GradScale learns per-layer scaling factors to optimally map gradients into the FP4 range before quantization ([43]).
  - *Microscaling (MXFP4):* Applying scaling factors at a fine-grained, per-group level within gradient tensors helps manage outliers and preserve precision for the bulk of the gradient values ([27]). The TetraJet paper proposes techniques like truncation-free scaling and double quantization to handle MXFP4 specifics effectively during training ([27]).

- **Mitigating Quantization Error/Bias:**
    - *Two-Phase Rounding (TPR):* This technique aims to cancel out quantization errors by using two slightly different FP4 rounding schemes (FP4-even and FP4-odd) for the two main gradient computations in backpropagation: the gradient of the loss with respect to layer inputs (dL/dx) and the gradient with respect to weights (dL/dW) ([35]).
    - *Logarithmic Unbiased Quantization (LUQ):* This method specifically targets unbiased 4-bit gradient quantization using a logarithmic format (E3M0). It combines stochastic rounding on a logarithmic scale for values within the representable range with stochastic mapping/pruning for underflow values and dynamic threshold adjustment to prevent overflow, ensuring the expected value of the quantized gradient matches the original ([35]).
    - *Differentiable Quantization Estimators:* As mentioned in QAT, using smooth, differentiable approximations of the quantization step can lead to better gradient flow during backpropagation, improving the accuracy of weight updates derived from FP4 computations ([12]).

3.3. Optimizers for Stability

Standard optimizers like Adam, while effective in higher precision, often exhibit instability when used with ultra-low precision formats like FP4 or INT4 ([46]). Key issues observed include heightened sensitivity to learning rate choices, unstable gradient norms (often exhibiting large spikes), loss divergence, and overall erratic training behavior ([46]). This has spurred research into optimizers specifically designed or adapted for low-precision training:

- **SPAM (Spike-Aware Momentum):** This optimizer introduces two main features: momentum reset (periodically resetting Adam's first and second moment estimates) and spike-aware gradient clipping (specifically clipping large gradient spikes). SPAM demonstrated improved performance over standard Adam in low-bit settings but remained sensitive to learning rate tuning and didn't fully eliminate gradient norm instability ([46]).
- **Stable-SPAM:** Building directly upon SPAM, Stable-SPAM incorporates two additional techniques to further enhance stability ([46]):
    - *Adaptive Spike-Aware Clipping (AdaClip):* Instead of a fixed clipping threshold, AdaClip adaptively updates the threshold based on the historical maximum values of observed gradient spikes.
    - *Adaptive Gradient Norm (AdaGN):* This technique normalizes the entire gradient matrix based on its historical l2-norm statistics before the optimizer update step. Combined with the momentum reset from SPAM, these additions

    were shown to effectively stabilize gradient norms during 4-bit LLM training, leading to superior performance and stability compared to both Adam and the original SPAM. Notably, Stable-SPAM reportedly allowed a 4-bit trained LLaMA-1B model to outperform a BF16 baseline trained with Adam ([46]).
- **Other Optimizers:** While not the primary focus, optimizers like Adafactor have shown surprising robustness to learning rate choices in low-bit settings, although potentially achieving lower peak performance than tuned SPAM variants ([46]).

3.4. Tackling Numerical Instability

Beyond gradient handling and optimizer choice, specific techniques target sources of numerical instability inherent in FP4 computations:

- **Outlier Handling:** Given the extreme sensitivity of FP4 to outliers ([12]):
    - *Clamping and Compensation:* One approach involves identifying activation outliers and clamping them to a maximum representable FP4 value to prevent overflow/underflow during quantization. To mitigate the information loss from clamping, compensation mechanisms using sparse auxiliary structures might be employed ([12]).
    - *Microscaling (MX):* As discussed, the per-group scaling nature of MX formats inherently limits the impact of outliers, making it a promising approach for managing activations and weights in FP4 ([3]).
- **Weight Oscillation Mitigation:** Training with quantized weights can sometimes lead to oscillations, where weights fluctuate back and forth across quantization boundaries between iterations. This was identified as a significant problem in MXFP4 training ([27]). Techniques proposed to address this include:
    - *EMA Quantizer (Q-EMA):* This method smooths the quantization process by basing the rounding decision not just on the current weight value but also on a slowly updated Exponential Moving Average (EMA) of the weight history ([27]).
    - *Adaptive Ramping Optimizer (Q-Ramping):* This technique identifies weights exhibiting frequent oscillations and adaptively reduces their update frequency (effectively increasing their local batch size and learning rate proportionally) to encourage them to move decisively away from quantization boundaries ([27]).

3.5. Strategic Mixed-Precision Training

Recognizing that full FP4 training across all operations might be too aggressive or unstable, especially for complex models, strategic mixed-precision approaches are often employed ([12]). The core idea is to use FP4 for the bulk of the computations (typically large matrix multiplies where the efficiency gains are highest) while retaining higher precision (e.g., FP8, BF16, or even FP32) for numerically sensitive components or operations.

Key considerations for mixed-precision FP4 training include:

- **Identifying Sensitive Components:** Research has highlighted specific parts of transformer models that are particularly sensitive to low precision:
    - *Attention Mechanism:* Computations involving the Query, Key, Value (QKV) projections and the final output projection within the multi-head attention block are often kept at FP8 to preserve the model's ability to learn attention patterns accurately ([16]). FP4 in attention can lead to near-uniform attention scores, hindering learning ([16]).
    - *Gradient Computations:* As detailed earlier, gradient calculations, especially for weights (dL/dW), often benefit from FP8 or higher precision due to range, underflow, and bias concerns ([16]). Activation gradients (dL/dx) may also require higher precision ([26]).
    - *Specific Layers:* In some architectures, certain layer types might be inherently more sensitive. For instance, 1x1 convolutions in ResNet residual blocks were found to require FP8 for maintaining accuracy in 4-bit training schemes ([43]).
- **Master Weights:** A common practice in mixed-precision training involves maintaining a master copy of the model weights in a higher precision format (e.g., FP32 or BF16). Gradients (potentially computed or accumulated in low precision) are used to update this master copy. The master weights are then quantized to the target low precision (FP4) for the forward and backward pass computations. This helps prevent the gradual degradation of weights due to repeated low-precision updates ([64]). This is likely essential for robust FP4 training.
- **Training Schedules:** Some successful FP4 training recipes employ multi-stage schedules. For example, a model might be trained predominantly using an FP4 mixed-precision scheme, followed by a shorter fine-tuning phase using a higher precision like FP16 or BF16 to help the model recover from quantization noise and converge to a better final state ([16]).

Table 3.1 summarizes the key techniques discussed for enabling native FP4 training.

Table 3.1: Key Native FP4 Training Techniques and Sources

Technique Category	**Specific Method**	**Core Idea**	**Key Challenge Addressed**	**Primary Source(s)**
QAT	Knowledge	Train FP4 model	Accuracy Loss	[18]

Adaptation	Distillation	to mimic higher-precision teacher/self		
	Differentiable Estimator	Use smooth approximation for quantization gradient	Gradient Mismatch, STE Inaccuracy	12
	Efficient QAT (e.g., Block-AP)	Train block-wise or only quantization parameters	QAT Computational Cost	18
	Coreset Selection (ACS)	Train on informative data subsets	QAT Data Cost, Robustness	31
Gradient Handling	Radix-4 FP4 / E3M0	Use formats with wider dynamic range for gradients	Gradient Range	35
	Adaptive Scaling (GradScale)	Learn per-layer gradient scaling factors	Diverse Gradient Ranges	43
	Microscaling (MXFP4)	Use fine-grained per-group scaling for gradients	Gradient Outliers, Range	27
	Two-Phase Rounding (TPR)	Use alternating rounding schemes for dL/dx and dL/dW	Quantization Error/Bias	35
	Logarithmic Unbiased (LUQ)	Combine logarithmic scale with stochastic	Quantization Bias, Gradient Distribution	35

		methods for unbiasedness		
Optimizer	SPAM	Momentum reset + spike-aware clipping	Stability, Loss Spikes	46
	Stable-SPAM (AdaClip + AdaGN)	Adaptive clipping + adaptive gradient normalization	Stability, LR Sensitivity, Divergence	46
Stability	Outlier Clamping/Compensation	Limit extreme activation values, potentially compensate	Activation Overflow/Underflow	12
	EMA Quantizer (Q-EMA)	Use weight EMA to guide rounding	Weight Oscillation	27
	Adaptive Ramping (Q-Ramping)	Reduce update frequency for oscillating weights	Weight Oscillation	27
Mixed Precision	Attention Protection	Use FP8 for sensitive attention computations (QKV, output)	Attention Accuracy/Stability	16
	Higher-Precision Gradients	Use FP8/BF16 for gradient computation (esp. dW)	Gradient Accuracy/Stability/Underflow	16
	Master Weights	Maintain higher-precision weight copy for	Weight Degradation	64

		updates		
	Phased Training Schedules	Finish training with a short higher-precision phase	Final Accuracy Recovery	16

4. Hardware Acceleration: The Role of Modern GPUs

The practical realization of FP4 training is intrinsically linked to the availability of hardware accelerators capable of efficiently executing 4-bit computations. While algorithmic techniques are essential for managing numerical stability and accuracy, dedicated hardware support is required to unlock the potential performance and efficiency benefits of FP4. Modern GPUs, particularly recent generations from NVIDIA, play a pivotal role in this ecosystem.

4.1. NVIDIA Blackwell Architecture (B200/GB200): Enabling FP4

The NVIDIA Blackwell architecture, introduced as the successor to Hopper, represents a significant leap forward in compute power and efficiency, explicitly designed to handle the demands of training and inferencing massive AI models, including trillion-parameter LLMs ([1]). Key features relevant to FP4 training include:

- **Architectural Design:** Blackwell GPUs feature a novel design comprising two large silicon dies connected via a very high-speed (10 TB/s) chip-to-chip interconnect, allowing them to function as a single, unified GPU ([2]). The GB200 Superchip combines two Blackwell GPUs with a Grace CPU ([1]). Large-scale systems like the GB200 NVL72 integrate 72 Blackwell GPUs into a single rack-scale, liquid-cooled unit connected via a high-bandwidth NVLink domain, designed to operate as one massive computational entity ([1]).
- **Native FP4 Support:** Blackwell is the first NVIDIA architecture to provide explicit, native hardware support for 4-bit floating-point (FP4) computations ([1]).
- **Second-Generation Transformer Engine:** This specialized hardware and software component is crucial for enabling efficient low-precision operations, including FP4 AI ([1]). It incorporates "micro-tensor scaling" techniques, likely referring to hardware support for microscaling (MX) formats, which use fine-grained, per-group scaling factors to optimize performance and accuracy, especially for FP4 ([3]). The engine dynamically manages precision and scaling to accelerate Transformer-based models while maintaining accuracy. It also supports FP8 precision, often highlighted for training speedups ([1]).
- **Fifth-Generation Tensor Cores:** These are the fundamental compute units

responsible for accelerating matrix multiply-accumulate (MMA) operations across various precisions. Blackwell's Tensor Cores are enhanced to provide significant acceleration for FP4 operations, offering a substantial increase in theoretical FP4 FLOPS compared to previous generations ([2]). The Blackwell Ultra variant further boosts FP4 compute performance ([3]).

- **Microscaling (MX) Format Support:** The hardware and the Transformer Engine are designed to efficiently handle MX formats, particularly MXFP4 (likely E2M1 elements with shared scaling factors), which helps mitigate the impact of outliers common in deep learning tensors ([3]).
- **Training vs. Inference Acceleration:** NVIDIA heavily promotes the inference capabilities of Blackwell with FP4, citing up to 30x speedups for real-time LLM inference compared to the Hopper H100 (using FP8) ([1]). However, the official documentation states that the Blackwell architecture, its Tensor Cores, and the second-generation Transformer Engine are designed to accelerate *both* training and inference ([1]). Software support reflects this, with frameworks like PyTorch, JAX, and TensorFlow being updated for Blackwell training and inference ([49]), and tools like TensorRT Model Optimizer supporting FP4 QAT ([49]). While FP4 acceleration is available for training, the most prominent training speedup claim for Blackwell (4x vs H100 at scale) is frequently associated with its enhanced FP8 capabilities and faster interconnects ([1]). The precise extent to which FP4 accelerates different parts of the training backward pass (e.g., gradient GEMMs) requires deeper technical specifications, possibly found in GTC presentations or detailed programming guides ([63]).

4.2. Comparison with Predecessors (Hopper H100)

The NVIDIA Hopper architecture (H100 GPU) was the first to introduce native support for 8-bit floating-point (FP8) formats (specifically E4M3 and E5M2) ([6]). This was enabled by its first-generation Transformer Engine and fourth-generation Tensor Cores ([6]). FP8 offered significant benefits over 16-bit formats, including up to 4x faster training for models like GPT-3 compared to the A100 ([6]) and substantial memory savings ([7]). Successful FP8 training relied on techniques like per-tensor dynamic scaling to manage its limited range ([7]).

Blackwell builds upon Hopper by adding native FP4 support, introducing the second-generation Transformer Engine with micro-tensor scaling, further increasing Tensor Core FLOPS (especially for FP4), and enhancing the NVLink interconnect bandwidth for better multi-GPU scaling ([1]).

Table 4.1 summarizes the key low-precision capabilities of these recent NVIDIA architectures.

Table 4.1: NVIDIA GPU FP8/FP4 Capability Comparison

Feature	**Hopper (H100)**	**Blackwell (B200/GB200)**
GPU Architecture	Hopper	Blackwell
FP8 Support (Native)	Yes (E4M3, E5M2)	Yes
FP4 Support (Native)	No	Yes (incl. MXFP4)
Transformer Engine Gen	1st Gen	2nd Gen
Key Enabling Tech (Low Prec)	Dynamic Scaling	Micro-Tensor Scaling (MX support)
Claimed Training Speedup	Up to 4x (vs A100, often FP8)	Up to 4x (vs H100, often FP8 + NVLink)
Claimed Inference Speedup	Up to 30x (vs A100, using FP8)	Up to 30x (vs H100, using FP4)
Relevant GPUs	H100	B200, GB200

4.3. How Hardware Facilitates FP4 Operations

Several hardware components work in concert to enable efficient FP4 computation:

- **Tensor Cores:** These specialized hardware units perform the core matrix multiply-accumulate operations at high speed for various precisions. Blackwell's fifth-generation Tensor Cores are specifically enhanced for FP4 throughput ([1]).
- **Transformer Engine:** This engine manages the complexities of using low-precision formats like FP4 and FP8, particularly within Transformer layers. It handles the selection of appropriate scaling factors (including fine-grained micro-tensor scaling for FP4) and dynamically adjusts precision to maximize performance while attempting to preserve accuracy ([1]).
- **Memory Subsystem:** High-bandwidth memory (HBM3e in Blackwell) is critical ([2]). Low-precision formats increase the computational intensity (more FLOPS per byte of data), demanding faster memory access to keep the Tensor Cores

supplied with data and avoid becoming memory-bound.
- **Interconnect (NVLink/NVLink Switch):** For training large models that span multiple GPUs, high-speed, low-latency interconnects are essential for exchanging activations and gradients. Blackwell features fifth-generation NVLink with increased bandwidth (1.8 TB/s per GPU) and supports larger NVLink domains (up to 576 GPUs), which helps manage communication overhead, especially relevant as computation speeds up with FP4 ([1]).

4.4. Future Hardware Outlook

The architectural progression from FP16/BF16 dominance to native FP8 (Hopper) and now native FP4 (Blackwell) indicates a clear industry trend towards exploring and supporting lower numerical precisions for deep learning ([23]). Future hardware generations may continue this trend, potentially exploring sub-4-bit formats, more exotic non-uniform representations, or formats specialized for specific operations like gradient accumulation. We can also anticipate increased flexibility and programmability in how low-precision formats and scaling mechanisms are handled, allowing for tighter co-design between algorithms and hardware architectures. The success of FP4 on Blackwell will likely influence the direction and investment in these future developments.

It is crucial to understand that while hardware like Blackwell provides the *capability* to perform FP4 computations at high speed ([1]), this capability alone does not guarantee successful or efficient FP4 *training*. As discussed in Section 3, naive application of FP4 leads to numerical catastrophe ([13]). Realizing the potential of FP4 training hinges on the sophisticated software ecosystem built upon this hardware. This includes specialized libraries (like NVIDIA's TensorRT-LLM, NeMo, cuDNN, and the Transformer Engine itself), integration into high-level frameworks (PyTorch, JAX, TensorFlow), and the implementation of advanced algorithmic techniques (Stable-SPAM, mixed-precision recipes, specialized gradient handling) ([3]). The impressive performance claims associated with FP4 often implicitly rely on the use of these highly optimized software stacks and recipes ([3]). Therefore, the hardware provides the potential, but it is the synergy between hardware, software, and algorithms that unlocks practical and effective FP4 training.

5. Performance Impact Analysis

The adoption of FP4 precision, particularly when coupled with hardware acceleration like that found in the NVIDIA Blackwell architecture, promises significant performance improvements across several dimensions, most notably in inference speed and overall efficiency. While training speedups are also anticipated, the most dramatic gains reported so far are concentrated on inference workloads.

5.1. Inference Acceleration

The use of FP4 for model inference yields substantial improvements in speed and throughput, enabling real-time responses from larger and more complex models.

- **Throughput Gains (Tokens/Second):** Hardware acceleration for FP4 matrix multiplication directly translates to faster token generation in LLMs.
    - NVIDIA reports that the GB200 NVL72 system delivers up to 30 times faster real-time inference for trillion-parameter LLMs compared to H100 systems (which typically use FP8 for inference) ([1]).
    - On the large DeepSeek-R1 (671B parameter) model, a single DGX B200 system (8x Blackwell GPUs) using FP4 precision achieved over 30,000 tokens per second maximum throughput and over 250 tokens per second per user ([49]).
    - Direct comparisons show DGX B200 (FP4) achieving over 3x the inference throughput of DGX H200 (FP8) on models like Llama 3.1 405B, Llama 3.3 70B, and DeepSeek-R1 ([49]).
    - The benefits extend beyond LLMs; for image generation using the Flux.1-dev diffusion model, FP4 inference yielded up to a 3x throughput increase (images per second) compared to FP16 ([49]).
- **Latency Reduction:** Increased throughput directly implies lower latency for generating each token ([1]). This is critical for interactive applications like chatbots and copilots, where responsiveness is key to user experience. The ability to perform computations faster with FP4 contributes significantly to reducing the time-to-first-token and subsequent token generation times ([69]).
- **Enabling Factors:** These inference speedups are attributed to the combination of higher raw FP4 FLOPS provided by the Blackwell Tensor Cores, optimizations within the second-generation Transformer Engine (including micro-tensor scaling), efficient software libraries like TensorRT-LLM and TensorRT Model Optimizer supporting FP4 PTQ and QAT outputs, and potentially reduced memory bandwidth bottlenecks due to smaller data sizes ([2]).

5.2. Efficiency Gains

Beyond raw speed, FP4 contributes to significant efficiency improvements in terms of memory usage and power consumption.

- **Memory Reduction:**
    - *Weights and Activations:* FP4 inherently requires half the storage of FP8 and a quarter of FP16/BF16 ([7]). This allows significantly larger models to fit into the

available GPU memory or reduces the number of GPUs required to hold a given model, lowering infrastructure costs ([3]).
  - *KV Cache:* In LLM inference, the Key-Value (KV) cache, which stores intermediate attention states, can consume substantial memory, especially for long input sequences. Quantizing the KV cache to low precision (potentially including FP4 or specialized INT formats) is a critical technique for enabling long-context inference efficiently ([18]).
  - *Example:* The FP4-quantized Flux.1-dev model demonstrated a 5.2x reduction in VRAM usage compared to its FP16 counterpart during inference ([49]).
- **Power Efficiency:**
  - NVIDIA claims the liquid-cooled GB200 NVL72 system offers up to 25 times better performance per watt (in terms of LLM inference throughput) compared to air-cooled H100 infrastructure ([1]). While liquid cooling contributes significantly to this, the inherent energy savings from performing computations and moving data using fewer bits with FP4 also play a role ([1]).

5.3. Training Speedup Potential

While FP4 offers clear advantages for inference, its impact on *native training* speed is more nuanced.

- **Theoretical Speedup:** On hardware like Blackwell that supports FP4 natively in its Tensor Cores, FP4 offers double the theoretical peak computational throughput compared to FP8, and four times that of FP16/BF16 ([7]). If training were purely limited by the speed of matrix multiplications, FP4 should provide substantial acceleration.
- **Observed Speedup & Comparisons:**
  - Experience with FP8 on Hopper showed significant training time reductions (e.g., up to 75% faster than BF16 for GPT-175B in some optimized frameworks) ([7]).
  - For Blackwell, NVIDIA's headline claim of 4x faster LLM training at scale compared to H100 is often attributed to the combination of enhanced FP8 capabilities and the faster fifth-generation NVLink interconnect ([1]). Specific, large-scale training speedup numbers directly attributed to using FP4 *instead of* FP8 or BF16 during training are less prominently featured in current materials compared to the inference claims.
  - Research papers focusing on native FP4 training techniques ([13]) primarily emphasize achieving accuracy comparable to BF16 or FP8 baselines ([12]). While achieving parity implies the *potential* for speedup due to lower precision, demonstrating substantial wall-clock time reductions for the entire training

process is often a secondary focus or requires further optimization.

- **Potential Bottlenecks:** Realizing the theoretical FP4 training speedup in practice depends on several factors:
    - *Compute vs. Other Bottlenecks:* Training speedup is only realized if matrix multiplication is the primary bottleneck. If other operations (data loading, non-GEMM computations, communication) dominate, FP4 acceleration will have less impact.
    - *Overhead of Techniques:* The complex techniques required for stable FP4 training (specialized scaling, gradient handling, stability checks, mixed-precision management) introduce computational overhead that can offset some of the gains from faster GEMMs.
    - *Software Optimization:* The efficiency of the software libraries (CUDA kernels, framework integration) implementing FP4 operations and the associated training techniques is critical.
    - *Mixed Precision:** If sensitive parts of the model remain in higher precision (FP8/BF16), the overall end-to-end speedup will be limited by those slower components.

Currently, the most heavily marketed and benchmarked performance gains for FP4 on architectures like Blackwell are concentrated in inference. Inference workloads are often simpler, more dominated by GEMMs, and quantization (PTQ or using a QAT-trained model) is a one-time offline cost. Training, conversely, involves the complexities of the backward pass, optimizer updates, and the need to maintain numerical stability throughout a long iterative process ([13]). The sophisticated techniques required for FP4 training add overhead, and achieving accuracy parity often requires careful tuning or specific training schedules ([26]). Therefore, while FP4 hardware exists and offers memory savings during training, realizing dramatic training *speedups* over already optimized FP8 or BF16 pipelines appears to be an area requiring further algorithmic refinement and software maturation. The primary driver for immediate FP4 adoption might thus be the significant inference cost and performance improvements it enables.

6. Accuracy, Quality, and Trade-offs

A primary concern when moving to ultra-low precision formats like FP4 is the potential degradation in model accuracy and quality. While FP4 offers compelling efficiency benefits, these must be weighed against its impact on the model's performance on its intended tasks. The goal of native FP4 training research is typically not just to make training possible, but to do so while achieving accuracy comparable to established higher-precision baselines like BF16 or FP8 ([12]).

6.1. Quantifying the Impact

Evaluating the impact of FP4 requires appropriate metrics and careful comparison against relevant baselines.

- **Metrics:** Common metrics vary depending on the model type and task:
    - *Language Models:* Perplexity (PPL) on validation datasets is a standard intrinsic measure. Accuracy on downstream NLP benchmarks like GLUE, SuperGLUE, or commonsense reasoning tasks (e.g., evaluating zero-shot performance) are extrinsic measures ([11]).
    - *Vision Models:* Top-1 and Top-5 accuracy on datasets like ImageNet ([27]).
    - *Generative Models (e.g., Diffusion):* Metrics evaluating image quality and text-image alignment, such as Fréchet Inception Distance (FID), CLIP score, or Human Preference Scores (e.g., HPSv2) ([22]). Human evaluation can also be crucial ([74]).
- **Reported Degradation:** The accuracy gap between FP4 and higher-precision baselines has narrowed significantly as techniques have improved:
    - *Early/Naive Attempts:* Initial experiments attempting 4-bit training without specialized techniques often resulted in substantial accuracy loss (e.g., 10-30% drop for ResNet using radix-2 FP4 gradients [43]). Using logarithmic FP4 gradients naively also caused severe degradation ([35]).
    - *Advanced Native FP4 Training (LLMs):* More recent work incorporating sophisticated techniques (mixed precision, specialized optimizers, gradient handling) frequently reports achieving "comparable accuracy" or "minimal degradation" relative to BF16 and FP8 baselines ([12]). The Stable-SPAM optimizer even enabled a 4-bit LLaMA-1B model to achieve lower (better) perplexity than a BF16 model trained with Adam ([46]).
    - *Advanced Native FP4 Training (Vision):* The TetraJet framework for MXFP4 training reduced accuracy degradation by over 50% compared to a baseline MXFP4 approach on Vision Transformers ([27]). Hybrid FP4/FP8 training reached within <1% of the FP32 baseline on ResNet50 ([43]). Using Adaptive Coreset Selection, 4-bit ResNet18 QAT achieved a 4.24% absolute accuracy gain over a random 10% data subset baseline ([31]).
    - *FP4 Post-Training Quantization (LLMs):* While PTQ generally incurs more accuracy loss than QAT at very low bits ([18]), methods like LLM-FP4 reported only a 5.8-point drop on LLaMA-13B zero-shot reasoning tasks, significantly outperforming previous 4-bit PTQ methods ([48]).
    - *FP4 Inference after QAT:* For inference deployment, FP4 QAT has shown

promising results, with NVIDIA reporting lossless FP4 quantization (compared to BF16 baseline) for their Nemotron-4 models using TensorRT Model Optimizer ([49]).

6.2. Factors Influencing Accuracy

The accuracy achieved with FP4 is not uniform but depends heavily on several factors:

- **Model Architecture and Size:** Different architectures and even different layers within the same model exhibit varying sensitivity to quantization ([11]). Attention mechanisms are known to be particularly sensitive ([16]). There is some evidence suggesting that larger models might be inherently more robust to quantization, potentially due to parameter redundancy ([11]).
- **Task Complexity:** Simpler tasks might tolerate FP4 with less degradation than highly complex tasks requiring fine-grained numerical distinctions.
- **Quantization Techniques:** This is arguably the most critical factor. The choice of FP4 format(s), scaling methods (per-tensor, per-group/MX), rounding schemes (stochastic vs. deterministic), gradient estimation and handling techniques (LUQ, TPR, GradScale), optimizer stability (Stable-SPAM), outlier mitigation (clamping, MX), weight oscillation control (Q-EMA, Q-Ramping), and the strategic use of mixed precision all significantly influence the final accuracy ([13]).
- **Training Data and Procedure:** The amount and quality of training data matter. Techniques like knowledge distillation or using data subsets (coresets) can impact the effectiveness of QAT ([18]). Training schedules, such as finishing with a higher-precision phase, can also help recover accuracy ([26]).

The strong dependence of FP4 accuracy on the applied techniques is a recurring theme. Early attempts using straightforward quantization resulted in poor performance ([42]). However, as researchers developed sophisticated methods specifically designed to counteract the numerical challenges of FP4—addressing bias ([42]), range limitations ([27]), instability ([27]), and sensitivity ([26])—the reported accuracy gap compared to higher-precision baselines has dramatically shrunk ([13]). This demonstrates that the "cost" of FP4 in terms of accuracy is not fixed but is rather a function of the mitigation strategies employed. Success hinges on adopting these state-of-the-art methodologies.

6.3. Summary of Reported Results

Table 6.1 provides a consolidated view of accuracy results reported in recent studies involving FP4 training or quantization.

Table 6.1: Summary of FP4 Training and Quantization Accuracy Results

Study (Source)	Model(s) Trained/Quantized	Task/Dataset	FP4 Approach	Comparison Baseline	Reported Accuracy Metric / Delta
Wang et al. (2025) ([12])	LLMs up to 13B	LLM Pretraining	Native FP4 (Differentiable Estimator, OCC, Mixed)	BF16, FP8	Comparable accuracy, minimal degradation
Sun et al. (2020) ([43])	ResNet18, ResNet50	ImageNet	Native FP4 (Radix-4, GradScale, TPR, Mixed FP4/FP8)	FP32	ResNet18: ~1% loss. ResNet50: <1% loss (with hybrid FP8) vs >4.5% loss (naive full FP4)
Chmiel et al. (2021) ([35])	ResNet, BERT	ImageNet, GLUE	Native FP4 (LUQ for gradients, INT4 forward)	FP32	State-of-the-art for full 4-bit training at the time, minimal overhead
Liu et al. (2025) ([16])	LLMs	LLM Pretraining	Native FP4 (Mixed FP4/FP8, Attention/Gradient Protect)	BF16, FP8	Comparable accuracy to BF16/FP8
Liu et al. (2025) ([46])	LLaMA-1B	LLM Pretraining	Native 4-bit (Stable-SPAM optimizer)	BF16 (Adam)	Outperforms BF16 Adam baseline by up to 2 PPL
Shao et al.	DeiT, Swin	ImageNet1K	Native MXFP4	MXFP4	Reduced accuracy

(2025) ([27])			(TetraJet + Q-EMA / Q-Ramping)	Baseline	gap vs FP by >50%. DeiT-B: 77.3% (Q-EMA) vs 74.5% (Baseline) vs 75.6% (FP)
Kwon et al. (2023) ([48])	LLaMA-13B	Commonsense Reasoning	PTQ FP4 (W4A4, Search-based params)	FP16	5.8 point drop vs FP16 (63.1 vs 68.9 avg score), 12.7 points better than previous PTQ SoTA
Liu et al. (2023) ([31])	ResNet-18	ImageNet1K	QAT 4-bit (Adaptive Coreset Selection - 10% data)	Random 10% Baseline	68.39% vs 64.15% Top-1 accuracy
NVIDIA Blog (2025) ([49])	Nemotron-4 15B/340B	LLM Inference	QAT FP4 (via TensorRT Model Optimizer)	BF16	Lossless FP4 quantization reported

(Note: Accuracy results are highly dependent on specific experimental setups and metrics. This table provides a high-level summary based on the source snippets.)

7. State-of-the-Art and Practical Considerations

Native FP4 training represents the cutting edge of low-precision deep learning. While significant progress has been made, particularly in the last couple of years, it remains an evolving field with ongoing research, practical challenges, and considerations for implementation.

7.1. Survey of Recent Research Breakthroughs (2023-2025)

The feasibility of native FP4 training has been largely enabled by several key research advancements, many published in top AI/ML conferences (NeurIPS, ICML, MLSys) or appearing as preprints on ArXiv ([11]). Notable contributions include:

- **Specialized FP4 Formats and Gradient Handling:** The work by Sun et al. (IBM) introduced the Radix-4 FP4 format, Adaptive Gradient Scaling (GradScale), and Two-Phase Rounding (TPR), demonstrating the importance of dynamic range and error cancellation for gradients ([43]). Chmiel et al. proposed Logarithmic Unbiased Quantization (LUQ), combining the theoretically optimal logarithmic scale for gradients (E3M0) with stochastic techniques to ensure unbiasedness ([35]).
- **Improved Quantization Simulation:** Wang et al. introduced differentiable quantization estimators for more precise weight updates and outlier clamping/compensation (OCC) strategies to handle activation extremes, forming a comprehensive FP4 training framework for LLMs ([12]).
- **Microscaling (MX) Format Training:** Shao et al. investigated training with the hardware-supported MXFP4 format, identified weight oscillation as a key issue, and proposed the TetraJet framework incorporating EMA Quantizer (Q-EMA) and Adaptive Ramping Optimizer (Q-Ramping) to mitigate it ([27]).
- **Stable Optimizers:** Liu et al. analyzed optimizer instability in 4-bit training and developed Stable-SPAM, significantly improving stability and performance through adaptive clipping and normalization techniques ([46]).
- **Mixed-Precision Strategies:** Liu et al. (different group) proposed effective mixed-precision recipes for FP4 LLM pretraining, identifying sensitive components like attention and employing higher precision (FP8) strategically alongside a target precision training schedule ([16]).

These breakthroughs collectively demonstrate that native FP4 training is achievable with near-baseline accuracy by employing carefully designed, multifaceted approaches.

7.2. Current Challenges and Open Questions

Despite the progress, several challenges and open research questions remain:

- **Generalization and Robustness:** How well do current FP4 training techniques generalize across diverse model architectures (beyond standard Transformers), datasets, and tasks? Ensuring robustness remains a key area.
- **Standardization:** The proliferation of different FP4 formats (E2M1, E3M0, Radix-4, MXFP4) and numerous training techniques makes standardization difficult. Establishing best practices and common recipes would aid adoption.
- **Ease of Use:** Implementing and debugging native FP4 training is complex, requiring deep expertise. Developing more user-friendly frameworks and tools that abstract away some of this complexity is needed.
- **Theoretical Understanding:** A deeper theoretical understanding of convergence guarantees, generalization properties, and the precise impact of quantization

noise in the ultra-low precision regime is still developing.

- **Optimizing the Full Pipeline:** While GEMMs are heavily accelerated, optimizing other parts of the training loop (e.g., normalization layers, activation functions, optimizer computations) for FP4 or ensuring seamless mixed-precision execution remains important.
- **Interaction with Other Techniques:** How does FP4 training interact with other efficiency techniques like network pruning/sparsity or Mixture-of-Experts (MoE) architectures? Exploring synergistic combinations is an area for future work.

7.3. Practical Steps and Recommendations for Implementation

For practitioners aiming to implement native FP4 training, based on current research and hardware capabilities, the following steps and considerations are recommended:

1. **Hardware Prerequisite:** Ensure access to hardware with native FP4 support, such as NVIDIA Blackwell GPUs (B200, GB200) or future equivalents.
2. **Software Ecosystem:** Utilize the latest versions of relevant software libraries provided by the hardware vendor (e.g., NVIDIA's CUDA, cuDNN, Transformer Engine, TensorRT-LLM, NeMo) and ensure compatibility with chosen deep learning frameworks (PyTorch, JAX, TensorFlow) ([3]).
3. **Adopt a Mixed-Precision Strategy:** Start conservatively. Do not attempt full FP4 quantization immediately. Begin by identifying potentially less sensitive components (e.g., weights in FFN layers) for FP4 quantization, while keeping known sensitive parts (e.g., attention mechanism computations, gradient accumulation, optimizer state) in a higher precision like FP8 or BF16 ([13]). Maintain master weights in FP32/BF16 ([64]).
4. **Employ Specialized Optimizers:** Use optimizers explicitly designed or validated for low-precision stability, such as Stable-SPAM, rather than standard Adam ([46]). Be prepared for careful learning rate tuning ([46]).
5. **Implement Advanced Gradient Handling:** Incorporate robust gradient scaling mechanisms. If using MXFP4 hardware, leverage its per-group scaling. Alternatively, consider adaptive per-layer scaling like GradScale ([27]). If possible and computationally feasible, explore unbiased quantization techniques inspired by LUQ ([35]).
6. **Address Stability Issues:** Implement techniques to handle activation outliers (e.g., via MX scaling or clamping/compensation) and potential weight oscillations (e.g., Q-EMA/Q-Ramping if using MX formats) ([13]).
7. **Monitor Training Dynamics Closely:** Diligently track loss curves, gradient norms (global and per-layer), accuracy metrics, and potential occurrences of NaNs or infinities. Instability often manifests as sharp spikes in loss or gradient norms ([46]).

8. **Leverage and Adapt Existing Recipes:** Study successful FP4 training frameworks and recipes reported in recent literature ([13]). Adapt these recipes to the specific model and task, starting with published hyperparameters where available.
9. **Consider QAT Fine-tuning:** If training from scratch proves too challenging or unstable, consider starting with a well-trained higher-precision model and applying FP4 QAT fine-tuning using appropriate techniques (e.g., knowledge distillation, efficient QAT methods) ([18]).

Implementing native FP4 training successfully requires a deep understanding of low-precision numerics, training dynamics, the specific algorithms being employed, and the underlying hardware and software stack. It is not a trivial "switch" from higher precision but involves careful integration and tuning of multiple specialized components ([12]). Debugging numerical issues can be significantly more challenging than in FP32 or BF16. Consequently, despite the availability of enabling hardware like Blackwell, native FP4 training currently remains largely the domain of expert researchers and practitioners in specialized labs. Widespread adoption will likely depend on the maturation of the software ecosystem and the development of more robust, automated, and user-friendly frameworks that encapsulate the necessary complexities.

8. Conclusion

8.1. Synthesized View

Native 4-bit floating-point (FP4) training represents a significant frontier in the quest for efficient deep learning, particularly for resource-intensive Large Language Models. Its feasibility has been established through a synergistic interplay between algorithmic innovation and dedicated hardware acceleration, exemplified by the capabilities of the NVIDIA Blackwell architecture. The primary allure of FP4 lies in its potential for substantial reductions in memory footprint, computational cost, and power consumption, alongside dramatic increases in inference throughput.

However, the journey to effective FP4 training is paved with significant numerical challenges stemming from its extremely limited dynamic range and precision. Overcoming these hurdles necessitates moving beyond standard training paradigms and embracing a sophisticated toolkit of specialized techniques. Successful native FP4 training typically requires:

- **Strategic Mixed Precision:** Applying FP4 judiciously while retaining higher precision (e.g., FP8, BF16) for numerically sensitive components like attention mechanisms and gradient accumulation.

- **Advanced Gradient Handling:** Employing specialized formats (e.g., Radix-4, E3M0), adaptive or fine-grained scaling (e.g., GradScale, MXFP4), and potentially unbiased quantization methods (e.g., LUQ) to ensure gradient integrity.
- **Stable Optimization:** Utilizing optimizers specifically designed to handle the instabilities of low-precision training, such as Stable-SPAM with its adaptive clipping and normalization features.
- **Targeted Stability Techniques:** Implementing methods to mitigate activation outliers and control weight oscillations.
- **Quantization-Aware Approaches:** Leveraging QAT principles, potentially combined with knowledge distillation or efficient training strategies like coreset selection, to help the model adapt to the constraints of FP4.

The success of FP4 is therefore not an inherent property of the 4-bit format itself, but rather a testament to the effectiveness of these co-developed algorithmic mitigation strategies. While hardware provides the essential computational substrate, it is the software frameworks, libraries, and carefully crafted training recipes that unlock practical FP4 training.

8.2. Concluding Remarks

FP4 precision marks a bold step towards ultra-low-precision deep learning, offering a compelling path to significantly improve the efficiency of training and deploying large-scale AI models. The dramatic inference speedups enabled by FP4 on architectures like Blackwell are already poised to impact the cost and responsiveness of AI services. While native FP4 training is now demonstrably feasible, achieving accuracy comparable to higher-precision baselines requires considerable expertise and careful application of advanced techniques, making it an expert task currently.

Future progress will likely focus on enhancing the robustness and ease-of-use of FP4 training methodologies, developing more standardized recipes, deepening the theoretical understanding of ultra-low precision learning, and potentially exploring even more aggressive quantization schemes. Continued co-evolution of algorithms, software frameworks, and hardware architectures will be essential to fully realize the potential of FP4 and further democratize the development and deployment of powerful AI models.

References

(Note: This section would typically contain a formatted list of all cited works. The

snippet IDs used throughout the text correspond to the provided research material.)

Appendix

(Optional: Could include detailed pseudocode for algorithms like Stable-SPAM, LUQ, Q-EMA; precise definitions of FP4 formats like E2M1, E3M0, Radix-4, MXFP4; derivations related to quantization error or gradient estimation.)

Works cited

1. GB200 NVL72 | NVIDIA, accessed April 28, 2025, https://www.nvidia.com/en-us/data-center/gb200-nvl72/
2. NVIDIA Blackwell GPUs: Architecture, Features, Specs - NexGen Cloud, accessed April 28, 2025, https://www.nexgencloud.com/blog/performance-benchmarks/nvidia-blackwell-gpus-architecture-features-specs
3. The Engine Behind AI Factories | NVIDIA Blackwell Architecture, accessed April 28, 2025, https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/
4. NVIDIA Blackwell Platform Arrives to Power a New Era of Computing, accessed April 28, 2025, https://nvidianews.nvidia.com/news/nvidia-blackwell-platform-arrives-to-power-a-new-era-of-computing
5. A deep dive into NVIDIA's Blackwell platform: B100 vs B200 vs GB200 GPUs, accessed April 28, 2025, https://blog.ori.co/nvidia-blackwell-b100-b200-gb200
6. H100 Tensor Core GPU - NVIDIA, accessed April 28, 2025, https://www.nvidia.com/en-us/data-center/h100/
7. FP8-LM: Training FP8 Large Language Models - arXiv, accessed April 28, 2025, https://arxiv.org/html/2310.18313v2
8. NVIDIA Hopper GPU Architecture, accessed April 28, 2025, https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/
9. FP8-LM: Training FP8 Large Language Models - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2310.18313
10. Nvidia's Breakthrough AI Chip Defies Physics (GTC Supercut) - Chaindesk, accessed April 28, 2025, https://www.chaindesk.ai/tools/youtube-summarizer/nvidias-breakthrough-ai-chip-defies-physics-gtc-supercut-odEnRBszBVI
11. scaling laws for mixed quantization in large language models - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2410.06722
12. [2501.17116] Optimizing Large Language Model Training Using FP4 Quantization - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2501.17116
13. Optimizing Large Language Model Training Using FP4 Quantization - arXiv, accessed April 28, 2025, https://arxiv.org/html/2501.17116v1
14. Optimizing Large Language Model Training Using FP4 Quantization - Hugging Face, accessed April 28, 2025, https://huggingface.co/papers/2501.17116

15. [Literature Review] Optimizing Large Language Model Training Using FP4 Quantization, accessed April 28, 2025, https://www.themoonlight.io/review/optimizing-large-language-model-training-using-fp4-quantization
16. Towards Efficient Pre-training: Exploring FP4 Precision in Large Language Models, accessed April 28, 2025, https://www.researchgate.net/publication/389091388_Towards_Efficient_Pre-training_Exploring_FP4_Precision_in_Large_Language_Models
17. A Survey of Low-bit Large Language Models: Basics, Systems, and Algorithms - arXiv, accessed April 28, 2025, https://arxiv.org/html/2409.16694v1
18. arXiv:2504.13932v1 [cs.LG] 14 Apr 2025, accessed April 28, 2025, https://www.arxiv.org/pdf/2504.13932
19. A Comprehensive Study on Quantization Techniques for Large Language Models - arXiv, accessed April 28, 2025, https://arxiv.org/html/2411.02530v1
20. arXiv:2502.12346v1 [cs.LG] 17 Feb 2025, accessed April 28, 2025, https://www.arxiv.org/pdf/2502.12346
21. arXiv:2502.12913v2 [cs.LG] 24 Feb 2025, accessed April 28, 2025, https://arxiv.org/pdf/2502.12913?
22. FP4DiT: Towards Effective Floating Point Quantization for Diffusion Transformers - arXiv, accessed April 28, 2025, https://arxiv.org/html/2503.15465v1
23. Learning from Students: Applying t-Distributions to Explore Accurate and Efficient Formats for LLMs - arXiv, accessed April 28, 2025, https://arxiv.org/html/2405.03103v2
24. Learning from Students: Applying t-Distributions to Explore Accurate and Efficient Formats for LLMs - GitHub, accessed April 28, 2025, https://raw.githubusercontent.com/mlresearch/v235/main/assets/dotzel24a/dotzel24a.pdf
25. Advances in the Neural Network Quantization: A Comprehensive Review - ResearchGate, accessed April 28, 2025, https://www.researchgate.net/publication/383437678_Advances_in_the_Neural_Network_Quantization_A_Comprehensive_Review
26. arXiv:2502.11458v1 [cs.LG] 17 Feb 2025, accessed April 28, 2025, http://www.arxiv.org/pdf/2502.11458
27. Oscillation-Reduced MXFP4 Training for Vision Transformers - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2502.20853
28. Towards Efficient Pre-training: Exploring FP4 Precision in Large Language Models - ar5iv, accessed April 28, 2025, https://ar5iv.labs.arxiv.org/html/2502.11458
29. Systematic Analysis of Low-Precision Training in Deep Neural Networks: Factors Influencing Matrix Computations - MDPI, accessed April 28, 2025, https://www.mdpi.com/2076-3417/14/21/10025
30. Learning from Students: Applying t-Distributions to Explore Accurate and Efficient Formats for LLMs - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2405.03103?
31. Efficient and Robust Quantization-aware Training via Adaptive Coreset Selection - arXiv, accessed April 28, 2025, https://arxiv.org/html/2306.07215v3

32. APTQ: Attention-aware Post-Training Mixed-Precision Quantization for Large Language Models - arXiv, accessed April 28, 2025, https://arxiv.org/html/2402.14866v2
33. arXiv:2502.11458v1 [cs.LG] 17 Feb 2025, accessed April 28, 2025, https://arxiv.org/pdf/2502.11458
34. Optimizing Large Language Model Training Using FP4 Quantization : r/LocalLLaMA - Reddit, accessed April 28, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1ictw5m/optimizing_large_language_model_training_using/
35. Accurate Neural Training with 4-bit Matrix Multiplications at Standard Formats - arXiv, accessed April 28, 2025, https://arxiv.org/html/2112.10769v4
36. arXiv:2504.05352v1 [cs.LG] 7 Apr 2025, accessed April 28, 2025, https://arxiv.org/pdf/2504.05352
37. Standalone 16-bit Neural Network Training: Missing Study for Hardware-Limited Deep Learning Practitioners - arXiv, accessed April 28, 2025, https://arxiv.org/html/2305.10947v5
38. DeepSeek-R1 and FP8 Mixed-Precision Training - Colfax Research, accessed April 28, 2025, https://research.colfax-intl.com/deepseek-r1-and-fp8-mixed-precision-training/
39. NVIDIA H100 GPU Specs and Price for ML Training and Inference — Blog - DataCrunch, accessed April 28, 2025, https://datacrunch.io/blog/nvidia-h100-gpu-specs-and-price
40. NVIDIA Hopper: H100 and FP8 Support - Lambda, accessed April 28, 2025, https://lambda.ai/blog/nvidia-hopper-h100-and-fp8-support
41. Using FP8 with Transformer Engine - NVIDIA Docs Hub, accessed April 28, 2025, https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/fp8_primer.html
42. arxiv.org, accessed April 28, 2025, https://arxiv.org/pdf/2112.10769
43. papers.nips.cc, accessed April 28, 2025, https://papers.nips.cc/paper/2020/file/13b919438259814cd5be8cb45877d577-Paper.pdf
44. AlphaTuning: Quantization-Aware Parameter-Efficient Adaptation of Large-Scale Pre-Trained Language Models - ACL Anthology, accessed April 28, 2025, https://aclanthology.org/2022.findings-emnlp.240.pdf
45. EfficientQAT: Efficient Quantization-Aware Training for Large Language Models - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2407.11062
46. Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam - arXiv, accessed April 28, 2025, https://arxiv.org/html/2502.17055v1
47. Floating-point arithmetic - Wikipedia, accessed April 28, 2025, https://en.wikipedia.org/wiki/Floating-point_arithmetic
48. LLM-FP4: 4-Bit Floating-Point Quantized Transformers - ACL Anthology, accessed April 28, 2025, https://aclanthology.org/2023.emnlp-main.39.pdf
49. NVIDIA Blackwell Delivers World-Record DeepSeek-R1 Inference ..., accessed April 28, 2025, https://developer.nvidia.com/blog/nvidia-blackwell-delivers-world-record-deepse

ek-r1-inference-performance/
50. [GTC 24 Keynote] Don't Miss This Transformative Moment in AI : r/nvidia - Reddit, accessed April 28, 2025, https://www.reddit.com/r/nvidia/comments/1bi0pyz/gtc_24_keynote_dont_miss_this_transformative/
51. Awesome-LLM-Compression/README.md at main - GitHub, accessed April 28, 2025, https://github.com/HuangOwen/Awesome-LLM-Compression/blob/main/README.md
52. Q4'24: Technology Update – Low Precision and Model Optimization - OpenVINO™ Blog, accessed April 28, 2025, https://blog.openvino.ai/blog-posts/q424-technology-update---low-precision-and-model-optimization
53. BlockDialect: Block-wise Fine-grained Mixed Format Quantization for Energy-Efficient LLM Inference - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2501.01144
54. BlockDialect: Block-wise Fine-grained Mixed Format for Energy-Efficient LLM Inference, accessed April 28, 2025, https://arxiv.org/html/2501.01144v1
55. Zhen-Dong/Awesome-Quantization-Papers: List of papers related to neural network quantization in recent AI conferences and journals. - GitHub, accessed April 28, 2025, https://github.com/Zhen-Dong/Awesome-Quantization-Papers
56. Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam - arXiv, accessed April 28, 2025, https://arxiv.org/pdf/2502.17055
57. [2502.17055] Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2502.17055
58. [2502.17055] Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam - ar5iv, accessed April 28, 2025, https://ar5iv.labs.arxiv.org/html/2502.17055
59. Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam | OpenReview, accessed April 28, 2025, https://openreview.net/forum?id=Gk2pBIAMUI
60. Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam - Hugging Face, accessed April 28, 2025, https://huggingface.co/papers/2502.17055
61. Efficient and Robust Quantization-aware Training via Adaptive Coreset Selection - OpenReview, accessed April 28, 2025, https://openreview.net/pdf?id=4c2pZzG94y
62. How far can we take quantization aware training (QAT)? : r/LocalLLaMA - Reddit, accessed April 28, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1k7rnu9/how_far_can_we_take_quantization_aware_training/
63. Think Fast: Inference Developers Conference Sessions | NVIDIA GTC 2025, accessed April 28, 2025, https://www.nvidia.com/gtc/sessions/think-fast-inference-developers/
64. "Optimizing Large Language Model Training Using FP4 Quantization", Wang et al. 2025, accessed April 28, 2025, https://www.reddit.com/r/mlscaling/comments/1ifzxjc/optimizing_large_language_model_training_using/

65. Nvidia CEO Jensen Huang full keynote at GTC 2024 - YouTube, accessed April 28, 2025, https://www.youtube.com/watch?v=f8DKD78BrQA
66. Nvidia's Breakthrough AI Chip Defies Physics (GTC Supercut) - YouTube, accessed April 28, 2025, https://www.youtube.com/watch?v=odEnRBszBVI
67. GTC March 2024 Keynote with NVIDIA CEO Jensen Huang - YouTube, accessed April 28, 2025, https://www.youtube.com/watch?v=Y2F8yisiS6E
68. NVIDIA GTC Keynote 2024 - Blackwell, NVLink Switch, NIM, Project GROOT - YouTube, accessed April 28, 2025, https://www.youtube.com/watch?v=CvOlsmiL8pk
69. Thousands of NVIDIA Grace Blackwell GPUs Now Live at CoreWeave, Propelling Development for AI Pioneers, accessed April 28, 2025, https://blogs.nvidia.com/blog/coreweave-grace-blackwell-gb200-nvl72/
70. NVIDIA Blackwell Ultra for the Era of AI Reasoning, accessed April 28, 2025, https://developer.nvidia.com/blog/nvidia-blackwell-ultra-for-the-era-of-ai-reasoning/
71. ArXiv Dives: The Era of 1-bit LLMs, All Large Language Models are in 1.58 Bits | Oxen.ai, accessed April 28, 2025, https://www.oxen.ai/blog/arxiv-dives-bitnet-1-58
72. GTC March 2025 Keynote with NVIDIA CEO Jensen Huang - YouTube, accessed April 28, 2025, https://www.youtube.com/watch?v=_waPvOwL9Z8
73. [2310.16836] LLM-FP4: 4-Bit Floating-Point Quantized Transformers - arXiv, accessed April 28, 2025, https://arxiv.org/abs/2310.16836
74. Q3'24: Technology Update – Low Precision and Model Optimization - OpenVINO™ Blog, accessed April 28, 2025, https://blog.openvino.ai/blog-posts/q324-technology-update---low-precision-and-model-optimization
75. NeurIPS 2024 Workshops, accessed April 28, 2025, https://neurips.cc/virtual/2024/events/workshop
76. MLSys 2024 Call For Papers, accessed April 28, 2025, https://mlsys.org/Conferences/2024/CallForPapers
77. ICML 2024 Papers, accessed April 28, 2025, https://icml.cc/virtual/2024/papers.html
78. ML with New Compute Paradigms Workshop at NeurIPS 2024, accessed April 28, 2025, https://www.mlwithnewcompute.com/
79. ML For Systems: Announcement, accessed April 28, 2025, http://mlforsystems.org/
80. HuangOwen/Awesome-LLM-Compression - GitHub, accessed April 28, 2025, https://github.com/HuangOwen/Awesome-LLM-Compression