INSTITUTE FOR PHYSICAL AI @ BMI
The Charlot Lab

Technical Report TR-2026-02
Survey / Review · Preprint v2
5 July 2026

MULTIPLY-FREE NEURAL COMPUTATION

Ternary and Low-Bit Neural Models, Their Training, and Their Silicon: A Survey

Restricting weights to $\{-1, 0, +1\}$ removes the hardware multiplier. This report reviews what that buys, how such models are trained, and where the silicon stands.

David Jean Charlot, PhD

Dean of Physical AI · The Charlot Lab, Institute for Physical AI @ BMI

Correspondence: contact@physicalai-bmi.org · physicalai-bmi.org · The Charlot Lab: labs.physicalai-bmi.org/charlot
Interactive companion: physicalai-bmi.org/research/charlot-lab#topic-ternary

Abstract. Multiplication is the most expensive arithmetic operation in a neural accelerator, and the multiplier array is the circuit block that benefits most from an advanced fabrication node. Ternary weight quantization constrains each weight to the set $\{-1, 0, +1\}$. Under this constraint a weight–activation product reduces to a sign selection and the multiplier is no longer required. This report reviews the resulting body of work. Section 2 states the review method and the first-order energy model used for the quantitative estimates. Sections 3 and 4 give the ternary representation and the quantization-aware training procedure that makes it accurate. Section 5 tabulates reported results, including a language model that reaches full-precision quality at 1.58 bits per weight and a vision-language-action policy that reports 11× lower memory and 4.4× lower latency. Section 6 develops the energy and storage accounting; under the model adopted here, replacing 32-bit floating-point weights with ternary weights reduces weight storage by 20.3× and removes 100% of multiplications. Section 7 surveys ternary-oriented silicon on 16–65 nm nodes. Section 8 states the current gap between available models and available hardware. The report reports no new experimental measurements; all figures cite prior work.

1. Introduction

Two quantities set the energy of a neural network evaluation: the cost of the arithmetic and the cost of moving operands between memory and the arithmetic units. Within the arithmetic, multiplication is the dominant term. Measured on a 45 nm process, a 32-bit floating-point multiply costs about 3.7 pJ and a 32-bit floating-point add about 0.9 pJ; an 8-bit integer multiply costs about 0.2 pJ against 0.03 pJ for the corresponding add[17]. The multiplier is also large and timing-critical. Because its speed and density improve most with each process generation, it is the block that most strongly pushes an accelerator toward a leading-edge, capital-intensive node.

These costs matter directly for embodied systems, which must run within the power and thermal budget of a body rather than a datacenter. A model that is affordable on a server may be inadmissible on a battery. The relevant design metric becomes energy per decision, and that metric rewards any method that removes multiplication. Ternary weight quantization is the most direct such method, and it is the subject of this report.

2. Scope and method

This is a review. It surveys published models, training methods, and hardware for ternary and closely related low-bit neural computation, drawing on peer-reviewed papers, preprints, and primary hardware disclosures from 1958 to 2026. Selection favored primary sources for each claim and the earliest source that established a method, so that the historical record is visible. The report states a research position in Section 9 but reports no original experiments.

Quantitative estimates use a first-order energy model. Per-operation energies are taken from Horowitz's 45 nm survey[17] and are treated as fixed coefficients; weight-storage energy is taken to scale linearly with the number of bits moved. The model counts multiply, add, and weight-fetch operations and ignores control, activation storage, interconnect, and utilization. It is therefore an order-of-magnitude accounting, adequate for comparing arithmetic families but not for predicting the performance of a specific chip. Two further limitations apply. First, the field moves quickly, and several hardware results cited here are pre-silicon simulations rather than measured parts; Section 7 marks this in each case. Second, reported model-quality figures are quoted from the originating papers under their own evaluation protocols and are not re-measured here.

3. Ternary weight representation

Let a linear layer compute $y = W \cdot x$ for a weight matrix W and activation vector x. In the full-precision case each output element accumulates products $W_{ij} \cdot x_j$, and each product is a multiply followed by an add:

$$E_{MAC} = E_{mul} + E_{add} \tag{1}$$

Ternary quantization restricts every weight to $w \in \{-1, 0, +1\}$. The product $w \cdot x$ then takes one of three values and requires no multiplier:

$$w \cdot x = \begin{cases} +x & \text{if } w=+1 \\ 0 & \text{if } w=0 \\ -x & \text{if } w=-1 \end{cases} \tag{2}$$

The canonical construction is absmean quantization[2]. A single scale γ is set to the mean absolute weight, each weight is divided by that scale and rounded to the nearest ternary value, and the scale is reapplied once at the output of the dot product:

$$\gamma = \frac{1}{n} \sum_i |W_i| \;; \quad W_q = \mathrm{clip}\left(\mathrm{round}(W / \gamma), -1, +1 \right) \;; \quad y = \gamma \cdot (W_q \cdot x) \tag{3}$$

The accumulation $W_q \cdot x$ uses additions and subtractions only. The single multiplication by γ is amortized over the whole dot product and is negligible for layers of realistic width. Weights whose magnitude falls below $\gamma/2$ round to zero, which introduces sparsity and removes the corresponding additions. A ternary value carries $\log_2 3 \approx 1.58$ bits of information, so the weight memory shrinks accordingly. Figure 1 contrasts the two datapaths.

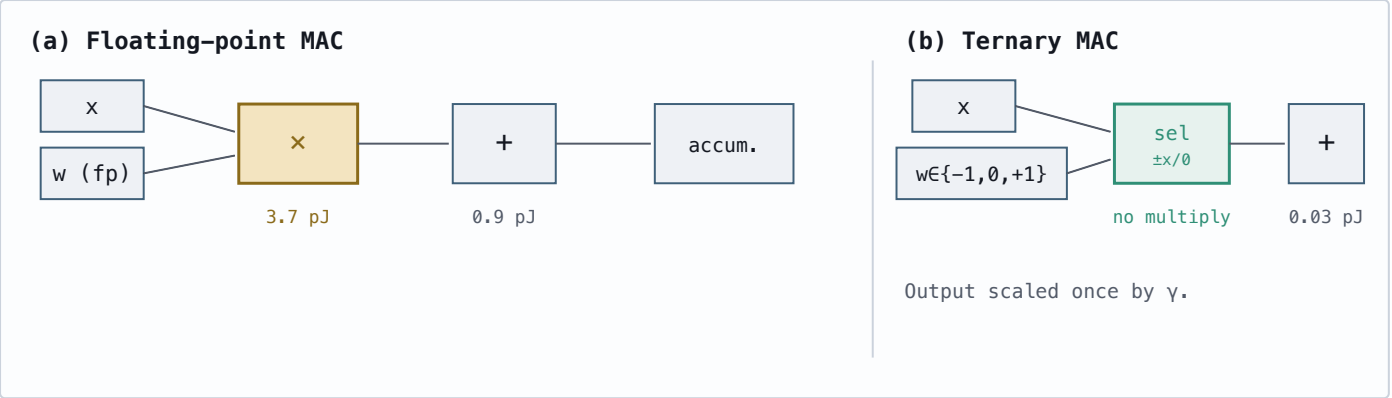

Figure 1. The multiply–accumulate datapath in floating point (a) and ternary (b). The multiplier, the shaded block in (a), is replaced in (b) by a three-way sign selection. Per-operation energies are the 45 nm figures of Horowitz[17].

4. Training at ternary precision

Rounding a network that was trained in floating point directly to ternary loses a large fraction of its accuracy. This loss is the reason ternary weights were set aside for many years. The remedy is quantization-aware training. The forward pass uses the ternary weights of equation (3), so the network learns parameters that already work after quantization. The backward pass keeps a full-precision copy of the weights, the shadow weights, and updates them through the non-differentiable rounding with a straight-through estimator, which passes the gradient through the quantizer as if it were the identity[9]:

$$\text{forward: use } W_q = \text{quant}(W) \quad ; \quad \text{backward: } \partial L/\partial W \leftarrow \partial L/\partial W_q \tag{4}$$

A network trained this way is ternary by construction rather than by compression. Ma and colleagues applied the procedure to transformer language models and reported that a model trained at 1.58 bits per weight matched a full-precision baseline of equal size on perplexity and on a suite of downstream tasks[2]. The point that is easy to miss is that ternary is a training-time decision, not a post-processing step. The same procedure underlies the earlier ternary weight networks[10] and trained ternary quantization[11], and the straight-through estimator itself predates the deep-learning quantization literature[9].

5. Reported results

Table 1 collects reported outcomes for trained low-bit models. The pattern across them is that model quality is retained while memory and latency fall by roughly an order of magnitude. For embodied control the relevant entry is BitVLA, a vision-language-action policy built on a 1-bit backbone, which reported parity with a full-precision baseline together with an 11× reduction in model memory and a 4.4× reduction in end-to-end latency[3]. Because the per-step cost of such a policy is dominated by the deterministic forward pass, these figures translate directly into battery life and thermal headroom on the robot.

Table 1. Reported results for trained low-bit models. Quality figures are as stated by the originating work under its own protocol.

MODEL	DOMAIN	WEIGHT PRECISION	QUALITY VS. FP	MEMORY	LATENCY	REF.
BitNet b1.58	language	ternary, 1.58 bit	matched (perplexity, downstream)	≈20× less*	lower	[2]

BitNet (1-bit)	language	binary	competitive	large reduction	lower	[5]
BitVLA	vision-language-action	1-bit backbone	matched baseline	11× less	4.4× lower	[3]

*Weight-storage estimate from the 32→1.58 bit ratio (Section 6), not a figure from [2].

6. Energy and storage accounting

Consider a layer with N weights, hence N multiply–accumulate operations per evaluation. Let b be the bits per weight and e_{bit} the energy to move one weight bit. Under the model of Section 2 the floating-point and ternary layer energies are

$$E_{fp} = N\cdot(E_{mul} + E_{add}) + N\cdot b_{fp}\cdot e_{bit} \tag{5}$$

$$E_{tern} = N_{nz}\cdot E_{add} + N\cdot b_t\cdot e_{bit}\ , \quad N_{nz} = (1-s)\cdot N \tag{6}$$

where s is the fraction of ternary weights that are zero. Two conclusions follow without choosing e_{bit}. The count of multiplications falls from N to zero. The weight storage falls by the bit ratio

$$b_{fp} / b_t = 32 / 1.58 = 20.3\times \tag{7}$$

Figure 2 places the per-operation energies on a common axis. The two features that drive the result are visible in it: a floating-point multiply is the most expensive arithmetic operation shown, and a memory access is more expensive still, so reducing both the arithmetic and the bits fetched compounds. Figure 3 shows the storage per weight across three precisions.

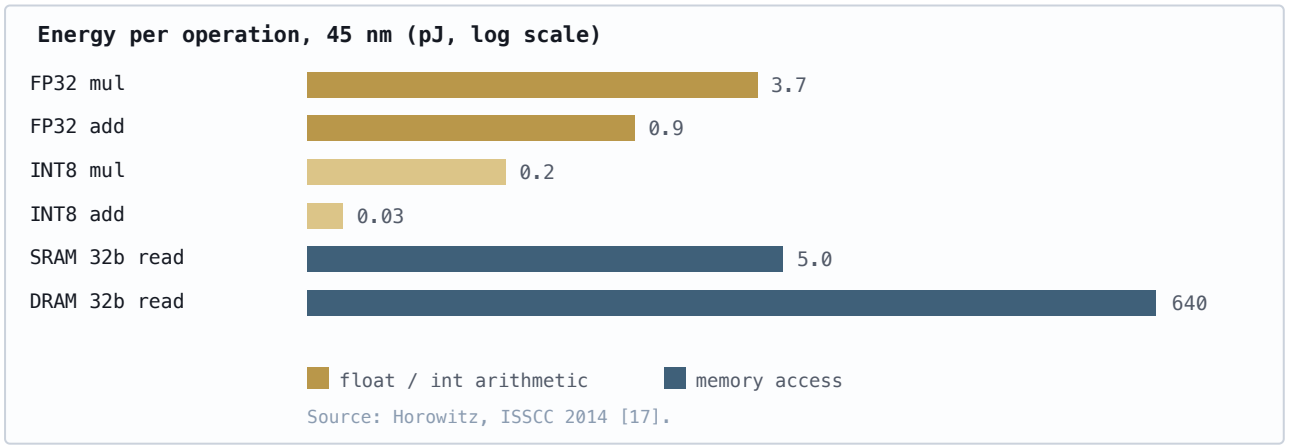

Figure 2. Energy of representative operations on a 45 nm process[17], log scale. A floating-point multiply is the costliest arithmetic operation shown, and a DRAM read exceeds it by more than two orders of magnitude, which is why removing both multiplications and fetched bits compounds.

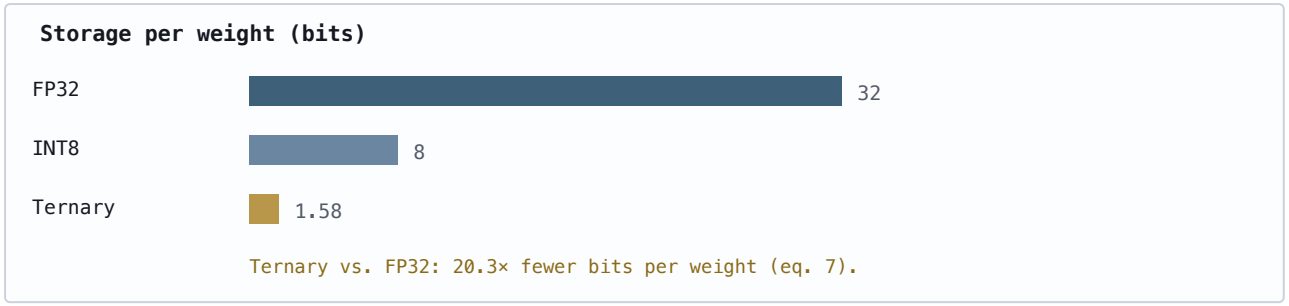

Figure 3. Weight storage per parameter at three precisions. Ternary uses $\log_2 3 \approx 1.58$ bits, a factor of 20.3 below 32-bit floating point.

Worked estimate. Take a layer of $N = 10^6$ weights and a measured ternary sparsity of $s = 0.3$[2]. Using the arithmetic coefficients of Figure 2 and counting arithmetic only, the floating-point layer spends $N \cdot (3.7 + 0.9) = 4.6\ \mu J$ per evaluation, while the ternary layer spends $N_{nz} \cdot 0.03 = 0.7 \cdot 10^6 \cdot 0.03\ pJ = 0.021\ \mu J$. The arithmetic energy ratio is about 220×, and every multiplication has been removed. The storage advantage from equation (7) is independent of this and applies to the fetch term.

7. Silicon

Additions and table lookups suit compute-in-memory, in which weights are stored in the array and the accumulation happens where they sit, so the memory–arithmetic boundary is not crossed for each operation. The idea dates to cellular logic-in-memory in 1969[18] and to Chua's prediction of the memristor in 1971[19]. Table 2 lists three recent ternary-oriented designs. They reach tens of TOPS/W on 16–65 nm nodes, which are mature, high-yield, and widely available. Removing the multiplier removes the main reason inference required a leading-edge process. The status column is important: the most fully multiply-free of the three, VitaLLM, is a post-layout simulation of a heterogeneous design in which a conventional integer core still performs attention, not a fabricated part.

Table 2. Representative ternary-oriented silicon.

DESIGN	NODE	APPROACH	REPORTED EFFICIENCY	STATUS	REF.
BitROM	65 nm	ROM compute-in-memory, 2 ternary weights/transistor	≈20.8 TOPS/W	research	[13]
TENET	28 nm	sparsity-aware LUT-centric ternary	≈1.5× speed, 1.6× energy	research, ASIC design	[14]
VitaLLM	16 nm	multiply-free TINT cores, $\mathrm{sel}(w,a) \in \{+a, 0, -a\}$	70 tok/s @ 66 mW	post-layout sim; heterogeneous	[15]

8. The logarithmic alternative, and the gap

Ternary is not the only route to multiply-free arithmetic. A logarithmic number system represents a value by its logarithm, so a multiply becomes an addition of exponents[6]; recent work approximates floating-point multiplication by an integer addition in the log domain with a small accuracy penalty[7]. Ternary quantizes the operand; the logarithmic method transforms the operation. Both remove the multiplier.

The state of the field can be stated plainly. Capable ternary models exist and continue to improve[2,4], and CPU inference kernels are available[8]. Ternary-native silicon is mostly at the research stage, as Table 2 shows. The logarithmic approach is earlier still, an algorithm ahead of its hardware, and the most

prominent logarithmic-number-system hardware effort withdrew from silicon to software in 2025[20]. The commercial mainstream has taken a different road, low-bit floating point: block-scaled microscaling formats[16] and vendor 8-bit formats[21] accept that the number format is the decisive variable but stop short of the ternary limit. Capable 1-bit models therefore arrived before hardware built to run them, and no fully ternary ecosystem yet ships.

9. Discussion

An embodied policy contains a deterministic path, the forward evaluation surveyed here, and a stochastic path, the sampling of actions and beliefs. Multiply-free arithmetic serves the deterministic path efficiently; a companion report treats the stochastic path and physics-based sampling. The two paths are complementary. The deterministic path performs no sampling and the stochastic path performs no matrix multiplication, so a system that runs each on the hardware suited to it need not pay for the operation it does not use. On this reading ternary is the natural arithmetic of the deterministic half of an energy-proportional edge system. The methods it rests on are open and long-published, from balanced ternary in 1958[1] to logic-in-memory in 1969[18] and the memristor in 1971[19].

10. Conclusion

Constraining weights to $\{-1, 0, +1\}$ turns multiplication into sign selection and removes the multiplier array. When the network is trained at ternary precision rather than compressed afterward, model quality is retained: reported results show parity with full precision in language and in embodied control, at roughly an order of magnitude less memory and latency. The first-order accounting of Section 6 gives a $20.3\times$ reduction in weight storage and the elimination of all multiplications. The supporting silicon runs on mature nodes but is still largely pre-production. For Physical AI at the edge, ternary arithmetic is the deterministic foundation on which an energy-proportional system can be assembled from open prior art.

References

1. N. P. Brusentsov et al. Setun: development and operation of a ternary computer. Moscow State University, 1958–65.
2. S. Ma, H. Wang, et al. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits (BitNet b1.58). arXiv:2402.17764, 2024; BitNet b1.58 2B4T Technical Report, arXiv:2504.12285, 2025.
3. H. Wang, S. Xiong, et al. BitVLA: 1-bit Vision-Language-Action Models for Robotics Manipulation. arXiv:2506.07530, 2025.
4. Sparse-BitNet. arXiv:2603.05168, 2026.
5. H. Wang, S. Ma, et al. BitNet: Scaling 1-bit Transformers for Large Language Models. arXiv:2310.11453, 2023.
6. J. N. Mitchell. Computer multiplication and division using binary logarithms. IRE Trans. Electronic Computers, 1962.
7. H. Luo et al. Addition is All You Need for Energy-efficient Language Models (L-Mul). arXiv:2410.00907, 2024.
8. J. Wang et al. bitnet.cpp: Efficient Edge Inference for Ternary LLMs. arXiv:2502.11880, 2025.
9. Y. Bengio, N. Léonard, A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432, 2013.
10. F. Li, B. Zhang, B. Liu. Ternary Weight Networks. arXiv:1605.04711, 2016.
11. C. Zhu, S. Han, H. Mao, W. J. Dally. Trained Ternary Quantization. arXiv:1612.01064, 2016.
12. M. Courbariaux, Y. Bengio, J.-P. David. BinaryConnect. NeurIPS, 2015; M. Rastegari et al. XNOR-Net. ECCV, 2016.
13. BitROM: Weight Reload-Free CiROM Architecture Towards Billion-Parameter 1.58-bit LLM Inference. arXiv:2509.08542, 2025.
14. TENET: An Efficient Sparsity-Aware LUT-Centric Architecture for Ternary LLM Inference on Edge. arXiv:2509.13765, 2025.
15. VitaLLM: A Versatile, Ultra-Compact Ternary LLM Accelerator with Dependency-Aware Scheduling. arXiv:2604.27396, 2026.

16. B. D. Rouhani et al. Microscaling Data Formats for Deep Learning. arXiv:2310.10537, 2023 (Open Compute Project MX specification).
17. Ascend HiFloat8 Format for Deep Learning. arXiv:2409.16626, 2024.
18. M. Horowitz. Computing's energy problem (and what we can do about it). IEEE ISSCC, 2014.
19. W. H. Kautz. Cellular logic-in-memory arrays. IEEE Trans. Computers, 1969.
20. L. O. Chua. Memristor—the missing circuit element. IEEE Trans. Circuit Theory, 1971.
21. Lemurian Labs raises $28M, pivots from LNS silicon to portability software. EE Times, Dec. 2025.

AI-use disclosure. Preparation of this report used a large language model (Claude, Anthropic) for drafting and editing text, organizing the reviewed literature, and preparing the figures and the interactive companion. Cited references were checked to resolve to their sources. The author reviewed the content and is solely responsible for it. Consistent with ICMJE, COPE, and IEEE guidance, the model is a tool and is not credited as an author.

Institute for Physical AI @ BMI · The Charlot Lab
503 McKeever Rd, Arcola, TX 77583, USA
physicalai-bmi.org · contact@physicalai-bmi.org

Technical Report TR-2026-02 · Preprint v2