Pairs or Bits: Two Repairs for the Binarized Convolution

Anonymous authors
Paper under double-blind review

Abstract

A binarized convolutional network computes each unit as a weighted count of input bits, one-bit activations against ternary weights. It deploys as a multiplier-free circuit and pays in accuracy, almost all of it charged to the activations and not the weights. Two repairs act inside the unit. One widens the activation to two or three bits, which widens every accumulator and the memory between layers. The other keeps one bit and lets each unit multiply its inputs in pairs as well as summing them, for four integer multiplies. The literature has developed the first. We develop the second and price both on what a circuit carries, logic area and the activation bits each layer passes on. We prove that the pair unit still compiles exactly, that neither repair's class contains the other, and that a rank condition on the pair-coefficient matrix, minimized over its diagonal, decides which degree-2 polynomials a rank-R unit realizes. Under one recipe for every model the pair term returns 2.11 of the 4.72 pp that binarization costs on CIFAR-10 at equal parameters, and 2.20 at equal logic. On that dataset and four more it returns between 1.46 and 2.11 pp while a second activation bit returns between 0.27 and 5.08, so which repair pays depends on the task. On two of the five the pair term leads while storing a third of the activation bits where the bit stores twice as many. It also reaches its plateau in a little over half the epochs, because a pair term gives the integer its threshold compares far more values to take. Neither the stored bits nor the speed depends on the learning rate. The accuracy gap does, and a tuned baseline closes it. Where activation bits are affordable, add them. Where memory or interconnect binds rather than arithmetic, add the pair term instead.

1 Introduction

A binarized convolutional network, with weights from a two- or three-valued alphabet, $\{-1, +1\}$ or $\{-1, 0, +1\}$, and one-bit activations, $\{0, 1\}$, is among the most aggressive quantizations that still train end to end (Hubara et al., 2016; Rastegari et al., 2016). Every multiply-accumulate becomes a bit operation and a count, weight memory shrinks by more than an order of magnitude, and the network can be deployed as a circuit rather than run as a program: each unit becomes a weighted count of its input bits, the number of bits its $+1$ weights select minus the number its -1 weights select, compared against a threshold, with no multiplier and no floating point. That is why binarized networks are a natural starting point for inference on FPGAs and other fixed-function hardware, and why a decade of work, from BNN (Hubara et al., 2016) and XNOR-Net (Rastegari et al., 2016) to Bi-Real (Liu et al., 2018) and ReActNet (Liu et al., 2020), has gone into recovering the accuracy they give up. Flows that compile such networks exist, FINN (Umuroglu et al., 2017) and LogicNets (Umuroglu et al., 2020), and their cost is not the multiply-accumulate: it is logic area and the activation bits each layer passes to the next. What binarization costs is accuracy, and it is the activations that cost it. Rastegari et al. (2016, Table 1) report 56.6 percent top-1 on ImageNet for full-precision AlexNet, 56.8 with binary weights, and 44.2 once the inputs are binarized as well; on our CIFAR-10 setup the one-bit network trails its float reference by 4.72 pp, with the same split (Appendix F). This paper measures how much of that two repairs of the unit recover and what each costs on those two measures, which is the comparison that decides between them for a deployment bound by activation memory or interconnect rather than by arithmetic.

We call a model hardware-aligned when its deployed form is a circuit of integer adders, comparators, logic gates and a bounded number of integer multipliers per unit, with no floating point, and its

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

deployed function is the one that was trained (Appendix F explains each term). Post-training quantization fails the second condition. Reaching a circuit requires training toward one, and two families do that. The first gives each unit an enumerated truth table (Petersen et al., 2022; Bacellar et al., 2024; Andronic & Constantinides, 2023), which caps fan-in at the table's size: one unit cannot read a 3×3 window over 128 channels, 1,152 bits, except by stacking levels. The second, the XNOR-Net and ternary-weight line (Rastegari et al., 2016; Li et al., 2016; Hubara et al., 2016), keeps the convolution and binarizes it in training with a straight-through estimator (Bengio et al., 2013); its unit counts rather than looks up, which lets it read the whole window in one level, and its forward pass is already discrete, so the compiled circuit computes the trained function. This paper works in the second family. There one unit already reads the whole window, and the open question is what it is allowed to do with what it reads.

The unit. One object carries the whole argument. A *unit* is what produces a single output bit: the weighted count above, taken over the whole window one filter sees. A layer is many units side by side and the network is layers of them, so the unit is the smallest piece that is repeated everywhere. It is worth looking at closely for two reasons. It is the piece that compiles to hardware, so its form fixes what the deployed circuit costs; and it is where the accuracy goes.

Two repairs. The accuracy binarization costs has two sources inside the unit. A weighted count takes a contribution from each input bit on its own, and nothing in it responds to two inputs being on together; and the unit passes on one bit of the integer it computed. The standard repair addresses the second, passing two or three bits per activation or a real-valued shortcut, the line from ABC-Net to ReActNet (Lin et al., 2017; Li et al., 2017; Liu et al., 2018; 2020; Bulat & Tzimiropoulos, 2019), at the price of widening every accumulator and the memory between layers. The other addresses the first, and is the one this paper develops: keep one-bit activations and let the unit multiply its inputs in pairs as well as summing them, reading the joint pattern of two channels while still passing one bit on. Its most general form gives every pair of the unit's n inputs its own coefficient, and n is 1,152 in the layers here, so a single unit would hold $\binom{n}{2} = 662{,}976$ of them and the widest layer 170M, more than thirty times what the whole network holds. We use a low-rank form instead: the weighted count plus a few products, each of two further weighted counts, equal on the cube to the pair form Figure 1 draws.

Three questions. Neither repair can express what the other can (Section 3), so the useful question is not which is better but what each costs. Does the lost accuracy come back at matched parameters and at matched logic? What does the pair term buy that a second activation bit does not, and what does each cost? And where does the gain stop? We answer all three in one codebase on five datasets, under a recipe fixed before any degree-2 model was trained, comparing at matched budgets on both of the costs a circuit carries.

Contributions. The binarized-network literature repairs the unit along one axis, precision. This paper adds a second, degree, and prices the two against each other on the costs the deployed circuit carries rather than in accuracy alone. The multiplicative unit itself is old (Rumelhart et al., 1986; Chrysos et al., 2020; Xu et al., 2024); what is new is putting it inside a binarized network and pricing it there, in one claim with three parts. **Mechanism.** A weighted count over one-bit activations takes a contribution from each input on its own, so degree and precision repair different things and neither expresses what the other does (Proposition D.1). **Remedy.** A low-rank pair term recovers part of what binarization costs and still deploys exactly, since $x_i^2 = x_i$ folds the squares away and BatchNorm folds into the threshold, at one integer multiply per product (Proposition 3.7); which functions a given number of products reaches is settled by a rank condition on the pair-coefficient matrix, minimized over its diagonal (Theorem 3.4, Corollary 3.5). **Measurement.** One codebase on five datasets, matched on parameters and on logic, with no accuracy of ours read against another paper's table. The pair term returns between $+1.46$ and $+2.11$ pp on every dataset, while a second activation bit returns between $+0.27$ and $+5.08$; so which repair pays depends on the task, and on two of the five it is the one storing a third of the bits.

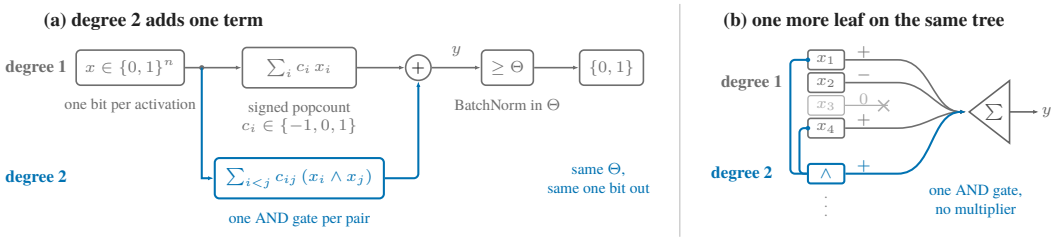

Figure 1: **Degree 2 adds one term to the same weighted count, and one leaf to the same adder tree.** Blue is everything degree 2 adds; gray is the binarized unit, unchanged. **(a)** The unit reads one-bit activations, forms a signed popcount with ternary coefficients, and thresholds the integer y with BatchNorm and the bias folded into Θ (Proposition C.1); degree 2 adds a second term over pairs, $x_i \wedge x_j$, into the same sum, so the threshold and the one-bit output are unchanged (Definition 3.1, Lemma 3.2). **(b)** Inside the count: a ternary coefficient is compiled-in wiring, $+$ routing an input into the positive count, $-$ into the negative and 0 dropping it, so the leaves cost no logic; the pair term is one more leaf of the same tree, formed by one AND gate over two inputs and carrying no multiplier. At fan-in $n = 1{,}152$ the tree is $\lceil \log_2 n \rceil = 11$ levels and y measures 11 to 20 bits, which a 24-bit accumulator discharges (Proposition 3.7). The pair form drawn here is the exact reference (Definition 3.6); at the fan-ins we train we deploy the algebraically identical factored form, which replaces the AND leaves by a handful of signed popcounts and one integer multiply per product (Section 3.5).

2 Background

Notation. A unit reads the flattened $k \times k$ patch of a convolution as $x \in \{0,1\}^n$ with $n = C_{\text{in}} k^2$, which is 1,152 at 128 input channels and $k = 3$; the width of a layer is its number of output channels. Every function on $\{0,1\}^n$ has a unique multilinear representation (O'Donnell, 2014), in which the pair term of Section 3 is analyzed; *generic* and *Lebesgue-null* are used in the usual sense over the real coefficients: a Lebesgue-null set has measure zero, so a coefficient vector drawn at random falls outside it with probability one. The network every concrete number refers to is that of Section 4.1.

2.1 Hardware-aligned networks, and how we count their cost

The first of the two families of Section 1 reaches a circuit by giving each unit an enumerated truth table: by hardening softmax-relaxed 2-input gates after training (Petersen et al., 2022; 2024), which leaves a discretization gap (Yousefi et al., 2025), by learning a t-input table directly (Bacellar et al., 2024; Wang et al., 2019), or by enumerating a small trained node into one (Andronic & Constantinides, 2023; 2024). All cap fan-in at t, as does LogicNets (Umuroglu et al., 2020).

A deployment flow for the second family, the one this paper works in, is FINN (Umuroglu et al., 2017), whose fold of BatchNorm and bias into one integer threshold is the degree-1 case of Section 3.3; its repairs are those of the next paragraph, and FracBNN (Zhang et al., 2021) is the partial precision repair, updating a sparse fraction of features at two bits; it also binarizes the input layer with a thermometer code, as our front end does. Published accuracies in this family are not comparators for Section 4: each differs from our setup in backbone, input encoding, first- and last-layer precision and recipe at once, so no cross-paper difference can be attributed to degree. On the last of those our stack is the stricter: its first layer reads binary planes from a frozen thermometer code and its readout admits an integer head (Section 4.2), where XNOR-Net leaves both in float. Appendix I gives the published numbers; every comparison in Section 4 changes one variable inside one setup.

Repairing the binarized convolution. Every repair in the literature addresses one loss. A binarized unit's pre-activation is a weighted count over the whole window, an integer of up to 12 bits, of which the activation keeps only the sign; that is the gray row of Figure 1. Each repair passes more of that integer on: several binary bases (Lin et al., 2017), a quantized residual (Li et al., 2017), real-valued shortcuts (Liu et al., 2018; 2020), learned per-channel scales (Bulat & Tzimiropoulos, 2019). What none of them adds is a term in which two channels multiply, so every unit in the family

stays additive across channels (Appendix D). The pair term is the other choice: it leaves the memory between layers alone and pays in logic and fan-out instead.

Logic and stored activation bits. Every network here carries two cost counts, and the two are never added. *Logic* is the area of the body, the layers between front end and readout, in LUT-equivalents, a LUT being the lookup-table cell an FPGA is built from; we count it *weight-shared*, one circuit per output channel reused across positions, from an analytical model rather than a synthesis result. *Stored activation bits* are what a layer writes for the next to read, summed over the body for one input image; this is traffic between layers, not the size of one memory, and it is not the parameter count, so networks matched on parameters need not be matched on stored bits. Appendix F gives both definitions, the two views, and what the model does and does not charge. We do not count bit operations. The BOPs metric charges a tap $b_a b_w + b_a + b_w + \log_2 nk^2$ at b_a activation and b_w weight bits, the last terms being the accumulator (Baskin et al., 2018); the simplified $b_a b_w$ in common use drops them. From one activation bit to three, at ternary weights and fan-in 1,152, those charge $1.40\times$ and $3.00\times$ against $1.67\times$ for the adder tree here, whose root already carries $\lceil \log_2 n \rceil$ bits. It is the simplified count that disagrees, and none of the three is a synthesis result.

2.2 Degree-2 operators: prior forms and expressivity

Units that multiply their inputs have a long history, and so do the objects they compute. A product of two linear forms is the sigma-pi unit (Rumelhart et al., 1986; Fan et al., 2020); its rank-factorized form is the Π-net (Chrysos et al., 2020), the asymmetric variant we adopt is QuadraNet's (Xu et al., 2024), and the same pair basis over binary features is the factorization machine (Rendle, 2010; Blondel et al., 2016). The degree-2 multilinear polynomials (O'Donnell, 2014) are exactly what our unit computes, which makes the rank question of Section 3.2 well posed. Our contribution is that operator on one-bit inputs with ternary factors, compiled exactly to an integer circuit and measured against the second activation bit. Concurrently, BiKAN (Fuad & Chen, 2026) adds a degree-2 term at one-bit precision with fixed circulant channel rolls in place of learned products; backbones differ, so we quote no cross-paper number, and Section 4.2's fixed-pair control is that comparison inside one setup.

A degree-2 threshold function is a polynomial-size polynomial threshold function, a class strictly inside depth-2 polynomial-size linear-threshold circuits (Bruck, 1990, Theorems 2.1 and 6.2) and strictly above one linear threshold gate (Proposition D.1(i)), so a degree-2 layer computes nothing such a circuit could not: cost selects it, not expressivity.

3 The degree-2 unit

The unit adds one term to the weighted count: a sum of products, each of two further weighted counts. The operator is not new (Section 2.2); its capacity on the cube and its exact compilation are what this paper adds. On one-bit inputs it collapses to a polynomial with one coefficient per input and one per pair, whatever factors produced it. Sections 3.1 to 3.3 assume one-bit activations, the setting of Section 4.2; Section 3.4 states what survives at two and three bits.

3.1 The unit and its realizations

A weighted count cannot see two inputs on together, and degree two is the smallest degree that can; Section 4.4 measures degree 3. A coefficient per pair is out of reach, as Section 1 counts it: 170M for the widest layer of the network of Section 4.1, against 5.1M for that whole network. The unit below sums R products instead, each multiplying two weighted counts of its inputs, storing $2R{+}1$ weight vectors rather than a coefficient per pair: at $R = 4$, nine vectors, a 64-fold reduction. Section 3.5 gives the deployed form.

Definition 3.1 (rank-R degree-2 unit)**.** *Fix $R \geq 1$ and parameters $\beta \in \mathbb{R}$, $w, a_r, b_r \in \mathbb{R}^n$ for $r = 1, \ldots, R$. The unit's pre-activation on $x \in \{0,1\}^n$, before BatchNorm (*BN*) and the activation, is*

$$g(x) \;=\; \beta + w^\top x + \sum_{r=1}^{R} (a_r^\top x)(b_r^\top x), \qquad \textit{and the unit outputs } \mathrm{act}(\mathrm{BN}(g(x))) \in \{0,1\}, \tag{1}$$

with $\mathrm{act}(t) = \mathbf{1}[t > 0]$. *With* $Q = \sum_r a_r b_r^\top$ *the pair term is* $x^\top Q x$, $\operatorname{rank} Q \le R$. *The unit* realizes *a polynomial* f *if* $g = f$ *on* $\{0,1\}^n$. *For a multilinear* f *of degree at most 2,* S_f *denotes its pair-coefficient matrix,* $(S_f)_{ij} = (S_f)_{ji}$ *the coefficient of* $x_i x_j$ *and* $(S_f)_{ii} = 0$.

We call R the rank and the $2R{+}1$ weighted counts the unit's *branches*, nine at $R = 4$. R is four unless the model's name says otherwise (Section 4.1), and factoring Q is the only structural constraint: the unit learns which weighted counts to multiply, not which pairs of inputs to keep.

The unit is a drop-in replacement for the binarized convolution: the $2R{+}1$ convolutions and their R products stand where one convolution stood, and BatchNorm, the activation and the one-bit interface between layers are untouched. Binarization had already replaced the float multiplier array by weighted counts; degree 2 adds R integer multiplies back. Parameters and training multiply-accumulates (MACs) both scale as branches times width squared, so at $R = 4$ width 64 matches degree 1 at width 192 in both ($9 \times 64^2 = 192^2$), to within 8 percent network-wide. Weights and activations are ternarized in the forward pass with a straight-through gradient and the readout is a float linear layer by XNOR-Net convention, an integer readout measured against it in Section 4.2 (Appendix I). From here on w, a_r, b_r and Q denote the ternarized tensors and $y = g(x) - \beta$ the integer the adder tree delivers to the comparison.

One identity fixes what the unit computes: an input equals its own square.

Lemma 3.2 (Booleanization). *On* $x \in \{0,1\}^n$, $x_i^2 = x_i$, *hence*

$$g(x) \;=\; \beta + \sum_i (w_i + Q_{ii})\, x_i + \sum_{i<j} (Q + Q^\top)_{ij}\, x_i x_j, \tag{2}$$

a multilinear polynomial of degree at most 2 with linear coefficients $c_i = w_i + Q_{ii}$ *and pair coefficients* $c_{ij} = (Q + Q^\top)_{ij}$. *Since the multilinear representation on the cube is unique, a rank-*R *unit realizes* f *iff there is* Q *of rank at most* R *with* $\operatorname{offdiag}(Q + Q^\top) = \operatorname{offdiag}(S_f)$.

The diagonal of Q lands on the linear coefficients, which w absorbs, so a target constrains the off-diagonal alone. With ternary factors every coefficient is a small integer, $|c_i| \le R{+}1$ and $|c_{ij}| \le 2R$, so Section 3.3 needs no rounding. Figure 1 draws the parameter and data streams.

3.2 Rank capacity

How many products does an exact target need? For real symmetric S let $n_+(S), n_-(S)$ count its positive and negative eigenvalues, $\operatorname{MaxIn}(S) = \max(n_+, n_-)$. Asymmetry is the first saving, and the lemma below makes it precise: a symmetric Q spends one product per nonzero eigenvalue, an asymmetric one covers a positive and a negative at once.

Lemma 3.3 (asymmetric splitting). *For every symmetric* S, $\min\{\operatorname{rank} Q : Q + Q^\top = S\} = \operatorname{MaxIn}(S)$*; the lower bound is classical (Kratzke et al., 1988, Theorem 1; Yabe, 2015, Lemma 3.1), attainability elementary (Appendix C); Kratzke et al. give the lower bound in graph form.*

The free diagonal is the second: adding a diagonal to the pair matrix moves only linear coefficients, which w absorbs, so the unit picks the cheapest.

Theorem 3.4 (rank capacity). *For a multilinear* f *of degree at most 2, with* S_f *its pair-coefficient matrix (Definition 3.1), define* $\rho(f) = \min_{D \text{ diagonal}} \operatorname{MaxIn}(S_f + D)$. *A rank-*$R$ *unit realizes* f *if and only if* $\rho(f) \le R$.

Scope. The theorem quantifies over real factors, so necessity transfers to deployed ternary units and sufficiency does not; and it measures the polynomial, not the Boolean function the unit emits.

Corollary 3.5 (capacity band; a $\rho = 1$ structure). *(a)* $\rho(f) \le \lfloor n/2 \rfloor$ *for every* f. *(b) With* q_n *the least integer such that* $\binom{q_n+1}{2} > n$, *if* $R \le (n - q_n)/2$ *the realizable set is Lebesgue-null in the class, so almost no* f *is realizable: at* $n = 1{,}152$, $q_n = 48$, *so* $R = 576$ *suffices for every* f *and all but a null set need* $R \ge 553$. *(c) For* $c \ne 0$ *the all-ones pair term* $c \sum_{i<j} x_i x_j$ *has* $\rho = 1$ *and equals* $c\binom{s}{2}$ *with* s *the input popcount, the number of ones among the inputs, a function of one scalar.*

Appendix E checks the constructive parts of these results numerically, by carrying out the construction and enumerating the cube rather than by searching for a rank. Of these, the fixed-matrix lower bound of Lemma 3.3 is classical and the genericity bound of Corollary 3.5(b) is a standard dimension count; the free diagonal of Theorem 3.4 is what is ours.

Does the threshold rescue the unit? The unit emits a sign, not a polynomial, so a cheaper g of the same sign pattern might do. Two results in Appendix D close that from opposite sides: almost all degree-2 threshold functions still need rank linear in n, so the sign buys nothing in general, yet one explicit family, an all-pairs term perturbed too little to cross a half-integer threshold, has ρ linear in n while a one-product popcount unit computes its threshold exactly.

Why $R = 4$, then. The trained $R = 4$ is 0.7 percent of the 576 that suffices for the whole class and far below the 553 almost every target needs, so this unit does not reach its target exactly and is not meant to: what it adds is a graded co-activation signal used approximately. Rank is therefore set by the budget, not by capacity: parameters scale as $(2R{+}1)w^2$, so every added product is paid for in width, and Section 4.2 measures which of the two the accuracy prefers, and whether the $\rho = 1$ structure of (c) carries any of the repair.

3.3 Exact compilation

Exact means the circuit returns, on every input, the bit the trained unit returned, with nothing refit. Two things stand in the way, both settled offline: BatchNorm is float, and the sum could overflow a fixed-width accumulator.

Write the trained layer's BatchNorm as $\mathrm{BN}(g) = \gamma(g - \mu)/\sigma + \beta_{\mathrm{BN}}$, four float statistics fixed once training ends. FINN (Umuroglu et al., 2017) folds them and the bias into one integer threshold at degree 1; Proposition C.1 extends that fold to the degree-2 unit, with no quantization error and for either sign of the scale. With $y = g(x) - \beta \in \mathbb{Z}$ the unit is $\mathbf{1}[y' \geq \Theta]$ for an offline integer Θ, where $y' = y$ at a positive scale and $-y$ at a negative one with every coefficient negated. The fold loses nothing because y is an integer: BatchNorm implies the real cut point $\theta = \mu - \beta - \beta_{\mathrm{BN}}\sigma/\gamma$, and rounding it up changes no output, so $\Theta = \lfloor\theta\rfloor + 1$ there. That integer is computed once, offline, from the four statistics above, which are then discarded: no float survives in the deployed unit. Proposition C.1 gives the other two cases with the proof, and panel (b) of Figure 1 draws the circuit this defines.

Definition 3.6 (pair-form circuit)**.** *Given integer* $\{c_i\}$, $\{c_{ij}\}$ *with active pairs* $P = \{(i,j) : c_{ij} \neq 0\}$ *and integer* Θ*:* $C(x) = \mathbf{1}[\sum_i c_i x_i + \sum_{(i,j)\in P} c_{ij}\,\mathrm{AND}(x_i, x_j) \geq \Theta]$.

Proposition 3.7 (exact compilation)**.** *Let* $M = \sum_i |c_i| + \sum_{i<j} |c_{ij}|$ *and build the circuit of Definition 3.6 from the coefficients of Lemma 3.2, negated when the BatchNorm scale is negative, and the threshold* Θ *of Proposition C.1, clamped to* $[-M, M{+}1]$*, which leaves* C *unchanged (the ends compile to constant gates, as does* $\gamma = 0$*). With an accumulator of width* $\omega \geq 1 + \lceil \log_2(1 + M)\rceil$ *it satisfies* $C(x) = \mathrm{act}(\mathrm{BN}(g(x)))$ *for every* $x \in \{0,1\}^n$.

Ternary factors bound the coefficients, $|c_i| \leq R{+}1$ and $|c_{ij}| \leq 2R$, so $M \leq (R{+}1)n + 2R\binom{n}{2}$, which is 5,309,568 at $n = 1{,}152$ and $R = 4$ and puts ω at 24. The bound is loose: on trained units the sum measures 11 to 20 bits, so the accumulator is sized by the worst case rather than by them.

3.4 The unit at two and three bits

The two- and three-bit networks of Sections 4.3 and 4.4 feed the same unit with b-bit activations, a uniform quantizer of the clamped BatchNorm output to 2^b levels, and Definition 3.1 is evaluated on that input unchanged. The factored form, the threshold fold and the parameter and logic matching all carry over, since none depends on b. What does not is Lemma 3.2, since $x_i^2 = x_i$ fails off the cube, so at $b \geq 2$ there is no pair form and no AND gate, only the factored form, and with it the rank theory of Section 3.2. What b adds is adder width and activation memory, which is what Section 4.3 compares against the pair term. Appendix F works each of these through, including the one caveat that the three-bit models' learnable quantizer range is not compiled exactly here.

Proposition D.1 (Appendix D) separates the two repairs without ordering them. A degree-2 unit on one-bit inputs computes the exclusive-or of two channels, which no additive unit computes at any activation width; and an additive unit with two or more bits computes a second threshold on one channel's magnitude, which no unit over the fixed one-bit code computes at any rank. Section 4.3 measures it.

Why a pair term should help most at one bit. A one-bit activation carries no magnitude, so an additive unit responds only to how many of its weighted inputs are on; what a window holds beyond that count is which inputs are on together, which is what a pair product reads. As bits are added that magnitude reaches the count itself and the two repairs overlap, so the pair term's return should fall with b; Section 4.3 measures the overlap and Section 4.4 the ceiling.

3.5 What the deployed circuit holds, and what it costs

Of the two 2-input products in the unit, only one costs a real AND. A ternary weight against an activation is wiring once the weight is a constant: $+1$ routes the input into the positive count, -1 into the negative, 0 drops it. The pair product has two runtime operands, so it is a real AND gate, one per active pair, and only in the pair form. The pair form of Figure 1(b) is $|P|$ AND gates, one signed adder tree over $n+|P|$ terms and one comparison, with no multiplier; the factored form is $2R$ signed popcounts, R products of those, one add and one comparison, so $(2R{+}1)n$ add-operations and R data-by-data multiplies, the only multiply in the body. The pair form is cheaper only below about $\frac{4}{3}Rn$ active pairs, under one percent at $n = 1{,}152$, and trained units are far denser, so we deploy the factored form and keep the pair form as the exact reference (Appendix F).

The cost model charges four elements: an adder tap (a tree leaf), a multiply, a b-bit quantizer and an AND, of which only the AND is definitional; Appendix F gives the other three and repeats every count at three and seven times the multiply charge. No comparison turns on it: the adder tree is over 97.5 percent of the cost of every one-bit degree-2 network on either view.

4 Experiments

4.1 Setup and protocol

Every model runs in one codebase on CIFAR-10, with four further datasets for replication: a frozen thermometer front end, five 3×3 blocks of widths $w, w, 2w, 2w, 4w$, ternary weights and a float linear readout except where stated, float reference 89.56 ± 0.10. The *shared recipe* is 120 epochs, crop and flip, three seeds and one learning rate of 10^{-3} fixed before any degree-2 run; accuracy is the *last-10* mean, the average over the final ten epochs, as mean $\pm$ sample standard deviation, no model is tuned against another, and a *direction* is a difference read for its sign only. Models are named by what varies, so `A1d2R4w64` is one activation bit, degree 2, four products, width 64, and degree 1 is `R0`. Appendices I and A give the protocol and every model.

4.2 One-bit activations: does the lost accuracy come back?

Table 1 holds the comparison. Binarizing the network of Section 4.1 costs 4.72 pp against its float reference. At matched parameters the pair term returns $+2.11$ pp of that loss at the 5.1M-parameter budget; at matched logic it returns $+2.20$ pp while holding one percent fewer parameters than that baseline, and $+2.00$ pp at the 5.1M budget with a 1.6 percent parameter deficit, a control that repeats on all five datasets; it stores a third of the activation bits. Two things qualify these numbers, both taken against baselines still gaining at 120 epochs: per-model tuning of the rate closes the matched-parameter gap and narrows the matched-logic one, leaving speed, steadiness and stored bits untouched; training both models out at twice the budget, where all four converge, leaves both gaps open at $+1.50$ and $+1.45$ pp (Appendix G). The controls of Appendix F exclude three competing explanations: the parameter difference, the baseline's larger fan-in and its wider integer alphabet.

The matched-parameter comparison is matched on logic too, to within 3 percent (Appendix I). The same difference appears on four further datasets (Table 1).

Speed and steadiness. Figure 2 draws both. Measured against each model's own final accuracy, the pair unit reaches 95 percent of its plateau at epoch 28 and the baseline at 49; training MACs are matched to within 8 percent, so epochs are a fair budget. Its curve is also the steadier: the two ranges of epoch-to-epoch movement do not overlap at any width or rate measured (Appendix G). Whether the gain is capacity or optimization is not separated by these runs. The steadiness has a measured cause: the integer a unit thresholds changes by at least 1 whatever the unit, but a pair term multiplies two counts, so within a quarter of a standard deviation of the threshold the untrained additive unit's

Table 1: The matched-parameter comparison on five datasets, with the same architecture in float for reference. Δ is degree 2 minus degree 1, from unrounded means. Neither width is tuned: the budget fixes one and the matching rule the other, and Table 2 holds two further matched comparisons. The cost columns hold across the three 32×32 datasets. Logic is weight-shared LUT-equivalents relative to the degree-1 row, the float row counted the same way with 8-bit multiply-accumulates to place it (Appendix F). ‡28×28 grayscale: 8 input planes rather than 24, so the cost columns do not transfer, though the stored-bit ratio is a third there too (Appendix A). Degree 2 reaches this reference on SVHN, Fashion-MNIST and KMNIST and degree 1 on none of them.

		cost of the deployed body				test accuracy (%)				
unit	name	weights	wt. mem.	logic	act. mem.	CIFAR-10	SVHN	CIFAR-100	Fashion‡	KMNIST‡
degree 1	`A1d1R0w192`	5.03M	1.20 MiB	1.00×	78 KiB	84.84 ± 0.04	91.47 ± 0.09	57.82 ± 0.27	91.33 ± 0.06	96.33 ± 0.13
degree 2	`A1d2R4w64`	5.11M	1.22 MiB	1.03×	26 KiB	86.96 ± 0.07	93.18 ± 0.10	59.87 ± 0.04	92.79 ± 0.09	97.92 ± 0.14
Δ, degree 2 minus degree 1						$+2.11$	$+1.71$	$+2.05$	$+1.46$	$+1.59$
degree 2	`A1d2R4w63`	4.95M	1.18 MiB	1.00×	26 KiB	86.84 ± 0.24	93.25 ± 0.01	59.89 ± 0.12	92.90 ± 0.03	97.90 ± 0.06
Δ, from *below* degree 1 on both counts						$+2.00$	$+1.78$	$+2.07$	$+1.57$	$+1.57$
float	`F32d1R0w192`	5.02M	19.15 MiB	105.8×	2.44 MiB	89.56 ± 0.10	92.84 ± 0.11	65.73 ± 0.14	92.62 ± 0.15	97.88 ± 0.13

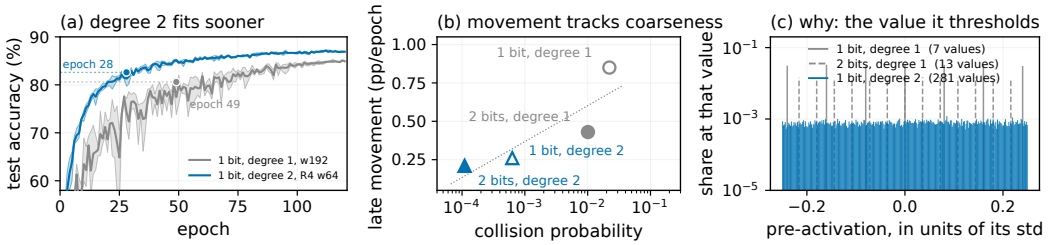

Figure 2: **Degree 2 fits sooner and moves less, and why.** CIFAR-10, the matched-parameter pair of Table 1. **(a)** Test accuracy against epoch, 3 seeds with the band giving their range; markers are where each curve first reaches 95 percent of its own final accuracy. **(b)** Late epoch-to-epoch movement of all four bit-and-degree configurations, measured in training, against their collision probability, measured on the untrained unit: the share of a unit's outputs that flip together when its threshold crosses one value. **(c)** That share drawn, also untrained, within a quarter of a standard deviation of the threshold: a comb of a few heavily loaded values for the additive unit, near-continuous once a pair term is added.

integer takes 7 distinct values and the pair unit's 281 (Figure 2(c)). One step of the threshold therefore reclassifies a far larger share of the inputs without a pair term than with one, about thirty times as large on trained models (Appendix B).

Without the float readout, and with fixed pairs. The deployed body has no floating-point element once head and readout are integer as well, and the edge survives there on all four ten-class datasets, $+1.61$, $+1.20$, $+1.42$ and $+1.11$ pp with disjoint seed bands; where it narrows, the baseline gains and the pair unit does not (Appendix F). Fixing the pair matrix to all ones and learning one scalar per channel returns $+0.05$ pp where the learned unit returns $+2.11$, so the gain needs learned pairs, not the extra convolution (Table 3).

Rank and fan-in. Full fan-in and four products were fixed before any model above was trained, and a grid with float weights measures both (Appendix H): fan-in is worth $+27.9$ pp from 9 to 216 inputs per unit, while the whole rank ladder from one product to 48 spans less than a single step along it.

4.3 A second activation bit: what does the pair term buy that the bit does not?

Adding a second activation bit is the precision ablation: it repairs the additive unit by returning part of the magnitude it discarded. The pair term's return falls to under a point on top of the second bit (Section 4.4 gives the full ordering), because what the two repairs add overlaps: together they fall

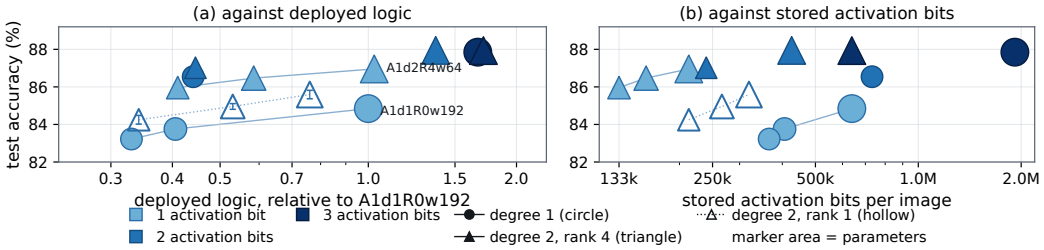

Figure 3: **The same models against the two cost counts.** CIFAR-10, 3 seeds. Lines join the one-bit width curves; only the two models the text names carry labels and Table 2 names every point; `A2d1R0w20` is below both axes. **(a)** Deployed logic on a logarithmic axis. **(b)** Stored activation bits per image. The second activation bit buys 0.58 pp more accuracy than the pair term and stores 5.5 times as many bits; that reversal is the exchange. Figure 4 in Appendix A adds what each repair buys on all five datasets, what each costs a unit, and the tabulated values.

1.73 pp short of the sum of their separate gains (Table 4). Once the activations themselves carry magnitude, part of what the pair term supplied reaches the additive unit directly, and less is left for the pair term to recover.

Every count of work calls the matched-parameter pair the same model, and only memory separates them (Appendix F). Both repairs recover accuracy; the question is the *exchange* between them, compared at *near-equal logic*, within 1.1 percent. Figure 3 places the models of Table 2 on both cost counts. The bit wins on accuracy, by 0.58 pp at one bit each, and the stored-bit ratio is what does not move; per-model tuning widens the bit's lead and leaves the ratio alone (Appendix G).

The two repairs load different parts of the circuit, which Figure 4(b), in Appendix A, separates at fixed width: a bit widens every adder tap and adds a quantizer, $1.34\times$ and $1.67\times$ the logic at two and three bits, and multiplies the activation memory by b; a pair term leaves that memory alone and multiplies the taps by $2R{+}1$. Bits also cost a signal line and a register stage each in a streaming design, so the pair term is the lever where memory or routing binds rather than multipliers (Appendix F).

4.4 Three bits, and degree 3: where does the gain stop?

Where degree stops mattering. On CIFAR-10 precision bounds degree and fan-in does not (both dials, Table 3): at matched parameters the return falls from $+2.11$ pp at one bit through $+0.21$ at two to $+0.09$ at three with overlapping bands, while the degree ordering holds at every fan-in measured. That bound is not general. On SVHN it runs the other way, $+1.71$, $+1.81$ and $+1.89$ pp at the same three widths, every model converged, because the bits buy the degree-1 model nothing there: precision bounds degree only where a bit has something to supply (Appendix A). Degree 3 scores 0.81 pp below degree 2 at matched parameters, but triples take a unit from nine weighted counts to 21, so matching costs it width and fan-in and it separates neither. Appendix H gives each in full.

5 Conclusion

Binarizing a convolution buys a multiplier-free circuit and pays in accuracy, almost all of it on the activations. The literature buys it back with bits, widening every accumulator and the inter-layer memory, which fixed-function deployment is shortest of. We repair the unit by degree instead, letting it multiply its inputs in pairs at one bit. The two are not substitutes: neither repair's class contains the other, a rank condition decides what a given number of products reaches, and the repaired unit still compiles exactly. At one bit a rank-4 pair term stores a third of the activation memory and reaches its plateau in a little over half the baseline's epochs, and returns about two points of the lost accuracy at matched parameters and at matched logic alike, with learned pairs and not merely an extra convolution. A tuned baseline closes that accuracy gap and closes nothing else: the stored bits, the speed and the exact deployment do not depend on the rate. Which repair to spend on is a property of the task rather than a default.

AI use statement

In this work, we used generative AI tools to implement methods, namely writing and debugging our experiment and analysis code (a task with required disclosure). We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

Ethics statement

This work trains image classifiers on the public CIFAR-10, CIFAR-100, SVHN, Fashion-MNIST and KMNIST benchmarks and proposes a unit for deploying them as circuits. It involves no human subjects, no personal data and no application-specific risk beyond those of image classification in general.

Reproducibility statement

The code will be released upon publication. It holds the one codebase every model runs in, the runner scripts for the configurations of Tables 2 and 3, the cost-model script with the self-test that reproduces its LUT-equivalent counts, the per-layer check that the factored and pair forms agree, and the enumeration of Appendix E. Runs made after the code began dumping its configuration record every flag in their result file; earlier runs are reproduced from their runner scripts. Section 4.1 gives the network, the thermometer front end, the recipe and the seed count. Appendix C gives the assumptions and the complete proofs.

References

Milad Alizadeh, Javier Fernández-Marqués, Nicholas D. Lane, and Yarin Gal. An empirical study of binary neural networks' optimisation. In *International Conference on Learning Representations (ICLR)*, 2019.

Marta Andronic and George A. Constantinides. PolyLUT: Learning piecewise polynomials for ultra-low latency FPGA LUT-based inference. In *International Conference on Field-Programmable Technology (FPT)*, 2023.

Marta Andronic and George A. Constantinides. NeuraLUT: Hiding neural network density in Boolean synthesizable functions. In *International Conference on Field-Programmable Logic and Applications (FPL)*, 2024.

Alan T. L. Bacellar, Zachary Susskind, Mauricio Breternitz Jr., Eugene John, Lizy K. John, Priscila M. V. Lima, and Felipe M. G. Franca. Differentiable weightless neural networks. In *International Conference on Machine Learning (ICML)*, volume 235, pp. 2277–2295, 2024.

Pierre Baldi and Roman Vershynin. The capacity of feedforward neural networks. *Neural Networks*, 116:288–311, 2019a.

Pierre Baldi and Roman Vershynin. Polynomial threshold functions, hyperplane arrangements, and random tensors. *SIAM Journal on Mathematics of Data Science*, 1(4):699–729, 2019b.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Chaim Baskin, Eli Schwartz, Evgenii Zheltonozhskii, Natan Liss, Raja Giryes, Alex M. Bronstein, and Avi Mendelson. UNIQ: Uniform noise injection for non-uniform quantization of neural networks. *arXiv preprint arXiv:1804.10969*, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. Higher-order factorization machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Jehoshua Bruck. Harmonic analysis of polynomial threshold functions. *SIAM Journal on Discrete Mathematics*, 3(2):168–177, 1990.

Adrian Bulat and Georgios Tzimiropoulos. XNOR-Net++: Improved binary neural networks. In *British Machine Vision Conference (BMVC)*, 2019.

Grigorios G. Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Yannis Panagakis, Jiankang Deng, and Stefanos Zafeiriou. Π-nets: Deep polynomial neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Fenglei Fan, Jinjun Xiong, and Ge Wang. Universal approximation with quadratic deep networks. *Neural Networks*, 124:383–392, 2020.

Kazi Ahmed Asif Fuad and Lizhong Chen. BiKAN: Restoring collapsed basis of binary Kolmogorov–Arnold networks. *arXiv preprint arXiv:2608.01490*, 2026.

David Money Harris and Sarah L. Harris. *Digital Design and Computer Architecture*. Morgan Kaufmann, 2nd edition, 2012.

Mark Horowitz. Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Thomas Kratzke, Bruce Reznick, and Douglas West. Eigensharp graphs: Decomposition into complete bipartite subgraphs. *Transactions of the American Mathematical Society*, 308(2):637–653, 1988.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Zefan Li, Bingbing Ni, Wenjun Zhang, Xiaokang Yang, and Wen Gao. Performance guaranteed network acceleration via high-order residual quantization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-Real net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm. In *European Conference on Computer Vision (ECCV)*, 2018.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. ReActNet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision (ECCV)*, 2020.

Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.

Steffen Rendle. Factorization machines. In *IEEE International Conference on Data Mining (ICDM)*, 2010.

David E. Rumelhart, Geoffrey E. Hinton, and James L. McClelland. A general framework for parallel distributed processing. In *Parallel Distributed Processing, Vol. 1*. MIT Press, 1986.

Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. FINN: A framework for fast, scalable binarized neural network inference. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017.

Yaman Umuroglu, Yash Akhauri, Nicholas J. Fraser, and Michaela Blott. LogicNets: Co-designed neural networks and circuits for extreme-throughput applications. In *International Conference on Field-Programmable Logic and Applications (FPL)*, 2020.

Erwei Wang, James J. Davis, Peter Y. K. Cheung, and George A. Constantinides. LUTNet: Rethinking inference in FPGA soft logic. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019.

Neil H. E. Weste and David Money Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley, 4th edition, 2010.

Chenhui Xu, Fuxun Yu, Zirui Xu, Chenchen Liu, Jinjun Xiong, and Xiang Chen. QuadraNet: Improving high-order neural interaction efficiency with hardware-aware quadratic neural networks. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 19–25, 2024.

Akihiro Yabe. Bi-polynomial rank and determinantal complexity. *arXiv preprint arXiv:1504.00151*, 2015.

Shakir Yousefi, Andreas Plesner, Till Aczel, and Roger Wattenhofer. Mind the gap: Removing the discretization gap in differentiable logic gate networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yichi Zhang, Junhao Pan, Xinheng Liu, Hongzheng Chen, Deming Chen, and Zhiru Zhang. FracBNN: Accurate and FPGA-efficient binary neural networks with fractional activations. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2021.

A The models, the controls and the exchange configurations

Table 2 lists every model the paper trained, Table 3 the controls and the readout variants, and Table 4 the configurations of the exchange with the gaps derived from them. Figure 3 gives the shape of the first of these; the table gives the values, the widths, the best-of-run accuracies and the unrolled cost view.

What each repair buys, across the datasets. Figure 4(a) puts the two repairs side by side on every dataset, each measured against that dataset's own one-bit degree-1 model at matched parameters. The pair term is the steadier of the two: it returns $+2.11$, $+1.71$, $+2.05$, $+1.46$ and $+1.59$ pp on CIFAR-10, SVHN, CIFAR-100, Fashion-MNIST and KMNIST, a spread of 0.65. A second activation bit returns $+2.84$, $+0.27$, $+5.08$, $+1.74$ and $+0.74$, a spread of 4.81, and on SVHN and KMNIST it returns less than the pair term does while storing twice the activation bits where the pair term stores a third as many. The ordering the main text reports, that the bit buys more accuracy than the pair term, is a CIFAR-10 ordering and it does not hold on two of the five. Two things keep this from being stronger than it is: every two-bit rung is still gaining at 120 epochs (Appendix G), so the bit's side is a lower bound and both reversals would narrow with budget; and this compares the bit on a degree-1 model with the pair term at one bit, not a degree-2 model at two bits, which exists on CIFAR-10 only.

The precision boundary is not general. Section 4.4 reports the pair term's return falling from $+2.20$ pp at one bit to $+0.48$ at two on CIFAR-10, which is the evidence that precision bounds degree. SVHN is the counterexample, at matched parameters and three seeds: the degree-1 model reaches 91.47 ± 0.09 at one bit and 91.54 ± 0.08 at two, the pair unit 93.18 ± 0.10 and 93.35 ± 0.05, so the degree edge is $+1.71$ and then $+1.81$ pp, both with disjoint seed bands. The second bit buys $+0.07$ and $+0.17$ there against $+2.78$ and $+1.06$ on CIFAR-10. Both two-bit models are converged, at residuals 0.63 and 0.11 against the band of 2.57 (Appendix G), so this is not a budget artifact. Against CIFAR-10's collapse of $+2.11$, $+0.21$, $+0.09$ (Section 4.4), SVHN runs $+1.71$, $+1.81$, $+1.89$: it grows. What bounds degree on CIFAR-10 is the second bit having something to supply; where it has little, as on SVHN, whose degree-1 model scores 91.47 ± 0.09 at one bit, 91.54 ± 0.08

Table 2: The models of the width, precision and exchange comparisons at the shared recipe (CIFAR-10, thermometer input, 120 epochs, crop and flip, shared learning rate 10^{-3}, 3 seeds). Weights are ternary on every model, so the name carries only what varies: `A` activation bits, `d` degree, `R` products per unit, `w` width. A degree-1 unit is the same unit with no products, hence `R0`; every name has the same four fields, and the rank field says which family the model is in. Rows are grouped into the comparisons themselves: each block states what is held equal, gives the degree-1 model first and the degree-2 model under it, and Δ is the second minus the first, which makes a block read down rather than looked up. A block with no Δ has no matched partner. Cost: weight-shared LUT-equivalent of the body relative to `A1d1R0w192` (unrolled view in parentheses); bits: stored activation bits per image. Accuracy is test accuracy in %, mean $\pm$ std, with the mean of per-seed best in parentheses. Every Δ comes from unrounded means, so it may differ in the last digit from the rounded values beside it.

model	unit	params	width	cost (unrolled)	bits	test acc	(best)	Δ
One bit, matched parameters at 5.1M. The headline comparison of Section 4.2.								
`A1d1R0w192`	degree 1	5,031,562	192	1.000 (1.000)	638,976	84.84 ± 0.04	(85.21)	
`A1d2R4w64`	degree 2	5,105,546	64	1.028 (1.098)	212,992	86.96 ± 0.07	(87.27)	$+2.11$
`A1d2R4w63`	degree 2, at a deficit on both counts	4,949,227	63	0.997 (1.066)	209,664	86.84 ± 0.24	(87.12)	$+2.00$
One bit, matched parameters at 1.7M. The same comparison at a third of the budget.								
`A1d1R0w110`	degree 1	1,664,970	110	0.330 (0.338)	366,080	83.22 ± 0.27	(83.70)	
`A1d2R1w64`	degree 2	1,704,842	64	0.342 (0.365)	212,992	84.25 ± 0.23	(84.61)	$+1.03$
One bit, matched logic. The same comparison priced by circuit area instead.								
`A1d1R0w122`	degree 1	2,044,242	122	0.406 (0.413)	406,016	83.76 ± 0.04	(84.09)	
`A1d2R4w40`	degree 2	2,024,570	40	0.410 (0.461)	133,120	85.96 ± 0.23	(86.23)	$+2.20$
One bit, our width curve. No degree-1 model is matched to these; they trace cost against accuracy.								
`A1d2R4w48`	degree 2	2,896,042	48	0.585 (0.644)	159,744	86.47 ± 0.16	(86.73)	
`A1d2R1w80`	degree 2	2,649,450	80	0.530 (0.556)	266,240	84.96 ± 0.15	(85.27)	
`A1d2R1w96`	degree 2	3,801,418	96	0.761 (0.787)	319,488	85.59 ± 0.22	(85.88)	
Two bits, matched logic. The precision repair, with degree on top of it.								
`A2d1R0w110`	degree 1	1,664,970	110	0.441 (0.451)	732,160	86.54 ± 0.14	(86.80)	
`A2d2R4w36`	degree 2	1,647,154	36	0.446 (0.509)	239,616	87.02 ± 0.19	(87.27)	$+0.48$
Two bits, matched parameters at 5.1M.								
`A2d1R0w192`	degree 1	5,031,562	192	1.334 (1.335)	1,277,952	87.73 ± 0.10	(87.92)	
`A2d2R4w64`	degree 2	5,105,546	64	1.370 (1.464)	425,984	87.94 ± 0.17	(88.18)	$+0.21$
Two bits, matched stored bits to `A1d2R4w40` above, which is what the Δ is against.								
`A2d1R0w20`	degree 1	59,730	20	0.016 (0.020)	133,120	74.37 ± 0.33	(75.69)	-11.59
Three bits, matched parameters at 5.1M.[†] The degree-1 model reaches the float ceiling of this stack.								
`A3d1R0w192`	degree 1	5,031,562	192	1.669 (1.670)	1,916,928	87.85 ± 0.19	(88.01)	
`A3d2R4w64`	degree 2	5,105,546	64	1.713 (1.830)	638,976	87.93 ± 0.24	(88.12)	$+0.09$

[†] every model here uses the fixed clamp, so the ladder moves one variable at every step. The three-bit pair was first run with a learnable activation range, five extra parameters that the one- and two-bit models cannot use; those runs are reported in Appendix G and are not the primary rows.

at two and 91.47 ± 0.01 at three, the boundary does not appear. The degree-2 models at two and three bits exist on CIFAR-10 and SVHN; the other three datasets carry the one-bit comparison and the degree-1 ladder only.

Uniform quantization, and why not just use it. *Uniform quantization* keeps the convolution and maps weights and activations alike onto 2^b evenly spaced levels of a fixed range, the same b for both, a single dial setting the precision of the whole network; the binarized family instead pins the weights to a small fixed alphabet and moves the activation width alone. A reader may reasonably ask why not simply turn that one dial rather than binarize and repair the unit. The comparison has to be made at a fixed activation width or it is not one: a b-bit uniform model against a one-bit pair unit charges one side for bits the other is not spending. Table 5 makes it at matched width on CIFAR-10 and SVHN, the two datasets where both degrees were trained at every width, each rung setting the weight alphabet and the activation to the same b at the width of the degree-1 baseline so that every row holds the parameters of Table 1. The pair unit is ahead at every rung there and stores a third as many bits: by $+2.01$, $+0.15$ and $+0.02$ on CIFAR-10 and $+1.58$, $+1.48$ and $+2.01$ on SVHN. Two properties do not depend on the dataset. From three bits up the uniform alphabet needs a multiply per tap, which the ternary one never does. And the ladder saturates: carried past the table on CIFAR-10, to four and eight bits, it reads 87.71 ± 0.12 and 87.85 ± 0.11 against 87.80 ± 0.07 at two, so everything from two

Table 3: Every control and readout model with its comparator: shared recipe, 3 seeds, test accuracy, Δ from unrounded means. The three-bit and degree-3 rows change degree at matched parameters, so width follows and fan-in with it; the all-ones row holds width fixed and changes only the pair structure; The readout rows replace the float linear head, leaving the body unchanged; the grid's eight cells were trained together in one directory, and its float-head linear-readout corner returns Table 1's CIFAR-10 pair to within 0.06, which is the check that it is the same setup. The replication rows repeat the fully-integer corner on the three further ten-class datasets, against the same-dataset rows of Table 1. The last row is CIFAR-100, where the count readout needs more channels per class than these widths leave and the ternary head with a linear readout is what deploys instead (Appendix F); it is the one row where the head moves the degree-2 model and not the degree-1 one.

control	models, degree 2 against degree 1	degree 2	degree 1	Δ
what turns the gain off				
all-ones pairs	all-ones Q w192 vs `A1d1R0w192`	84.90 ± 0.18	84.84 ± 0.04	$+0.05$
three bits	`A3d2R4w64` vs `A3d1R0w192`	87.93 ± 0.24	87.85 ± 0.19	$+0.09$
degree 3	`A1d3R4w42` vs `A1d2R4w64`	86.14 ± 0.11	86.96 ± 0.07	-0.81
what the gain survives: the readout grid, all eight cells trained together, `A1d2R4w64` against `A1d1R0w192`				
float head, linear readout	the paper's default	86.96 ± 0.08	84.78 ± 0.08	$+2.18$
float head, count readout	the same bodies	86.88 ± 0.13	85.28 ± 0.22	$+1.60$
ternary head, linear readout	the same bodies	86.92 ± 0.18	85.56 ± 0.07	$+1.35$
ternary head, count readout	no float anywhere	86.91 ± 0.29	85.29 ± 0.23	$+1.61$
the same no-float-anywhere corner on the replication datasets				
ternary head, count readout	SVHN	93.05 ± 0.08	91.85 ± 0.13	$+1.20$
ternary head, count readout	Fashion-MNIST	92.81 ± 0.10	91.40 ± 0.14	$+1.42$
ternary head, count readout	KMNIST	97.98 ± 0.07	96.87 ± 0.03	$+1.11$
CIFAR-100, where the count readout cannot run: the head alone is what deploys				
ternary head, linear readout	CIFAR-100	58.79 ± 0.27	58.17 ± 0.16	$+0.62$

Table 4: The pair term's return at each activation precision, and what each repair stores. Rows are grouped by activation bits; within a group the degree-2 row is compared with the degree-1 row above it ("pair term"), and every row is compared with the one-bit binarized base ("vs base"). The one- and two-bit groups are at near-equal logic (within 1.1 percent inside each group; the two-bit models carry 8.7 percent more than the one-bit models, worth about 0.1 pp); the three-bit group is matched on parameters, not logic, since that is the configuration that locates the ceiling (Section 3.4). The three comparisons below the rule are the exchange between the repairs at two bits; the interaction is how far the two repairs together fall short of their separate sums, from unrounded means. Bit counts are exact and the ratios follow from them. All accuracies are at the shared recipe.

bits	model	degree	acc	pair term	vs base	stored bits (vs base)
1	`A1d1R0w122`	1, the binarized base	83.76	–	–	406,016 ($1.00\times$)
1	`A1d2R4w40`	2	85.96	$+2.20$	$+2.20$	133,120 ($0.33\times$)
2	`A2d1R0w110`	1	86.54	–	$+2.78$	732,160 ($1.80\times$)
2	`A2d2R4w36`	2	87.02	$+0.48$	$+3.26$	239,616 ($0.59\times$)
3	`A3d1R0w192`	1	87.85	–	$+4.09$	1,916,928 ($4.72\times$)
3	`A3d2R4w64`	2	87.93	$+0.09$	$+4.17$	638,976 ($1.57\times$)
the second bit over the pair term, at one bit each				$+0.58$		$5.5\times$ as many
the pair term on top of that bit				$+0.48$		$0.33\times$ as many
interaction of the two repairs				-1.73		

bits to eight spans 0.21 pp and the float reference still sits 1.65 above the highest rung. The three-bit pair unit passes that eight-bit rung, 87.93 against 87.85, while storing an eighth of its activation bits.

Two further configurations, for completeness. at equal stored bits `A2d1R0w20` lands 11.6 pp behind `A1d2R4w40` at one twenty-sixth of the logic, and at matched parameters the second bit adds 0.99 pp to degree 2, a direction measured at saturation. Proposition D.1 separates the two repairs in

Table 5: Uniform quantization at one, two and three bits, against the two models of Table 1 and the float reference. Rows share the architecture, the width and the shared recipe, and differ only in the weight alphabet and the activation width, both of which move together as uniform quantization requires. Weight memory is the body held once and activation memory one image's activations passing through it. The logic column of Table 1 is absent here on purpose: the cost model charges ternary-weight units, whose taps are signed adds, and from three bits up the alphabet needs a multiply per tap, which it does not price. The table is restricted to the two datasets where both degrees were trained at every activation width, because a comparison between a b-bit uniform model and a one-bit pair unit is not one at matched width; the paragraph above gives what the other three datasets show. The uniform ladder stops at two bits. Differences quoted in the paragraph come from unrounded means and may therefore differ in the last digit from the rounded values here. Every uniform rung above one bit is still gaining at this budget (Appendix G), so those rungs are lower bounds.

		cost of the deployed body			test accuracy (%)	
unit	weights	params	wt. mem.	act. mem.	CIFAR-10	SVHN
uniform 1 bit	$\{-1, +1\}$	5.03M	0.60 MiB	78 KiB	84.95 ± 0.18	91.60 ± 0.22
uniform 2 bits	$\{-1, 0, +1\}$	5.03M	1.20 MiB	156 KiB	87.80 ± 0.07	91.86 ± 0.08
uniform 3 bits	$\{-3, \dots, 3\}$	5.03M	1.80 MiB	234 KiB	87.92 ± 0.26	91.35 ± 0.14
degree 1, 1 bit	$\{-1, 0, +1\}$	5.03M	1.20 MiB	78 KiB	84.84 ± 0.04	91.47 ± 0.09
degree 2, 1 bit	$\{-1, 0, +1\}$	5.11M	1.22 MiB	26 KiB	86.96 ± 0.07	93.18 ± 0.10
degree 2, 2 bits	$\{-1, 0, +1\}$	5.11M	1.22 MiB	52 KiB	87.94 ± 0.17	93.35 ± 0.05
degree 2, 3 bits	$\{-1, 0, +1\}$	5.11M	1.22 MiB	78 KiB	87.93 ± 0.24	93.36 ± 0.12
float	float32	5.02M	19.15 MiB	2.44 MiB	89.56 ± 0.10	92.84 ± 0.11

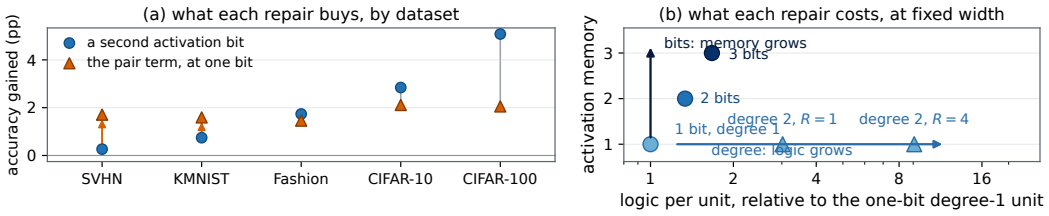

Figure 4: **What each repair buys, and what it costs. (a)** Accuracy gained over each dataset's one-bit degree-1 model at matched parameters, by a second activation bit and by the pair term, 3 seeds each. Each gain is taken within its own family, so no difference crosses quantizer classes: the bit is the uniform ladder's two-bit rung minus its one-bit rung at width 192, $+2.84$, $+0.27$, $+5.08$, $+1.74$ and $+0.74$ on CIFAR-10, SVHN, CIFAR-100, Fashion-MNIST and KMNIST; the pair term is Table 1's Δ. The pair term returns between $+1.46$ and $+2.11$ pp on all five datasets while the bit runs from $+0.27$ to $+5.08$, and on SVHN and KMNIST the bit falls below it, arrowed. The two-bit rungs are still gaining at this budget (Appendix G), so the bit's side is a lower bound and both reversals would narrow with more epochs. **(b)** What each repair costs a unit at fixed width, from the cost model of Section 3.5: bits multiply the activation memory, degree multiplies the adder taps. Points are labeled in place; the legend of Figure 3 applies to panel (b).

what they can express; accuracy at this operating point does not, because what each adds is largely shared, or compressed by the ceiling.

B How coarsely each unit's threshold input is resolved

This appendix carries the mechanism behind the speed and steadiness of Section 4.2: how finely the integer a unit thresholds is resolved, simulated on the unit, measured inside trained networks, and traced along the depth of the stack.

How coarsely each unit's threshold input is resolved. The threshold of Proposition 3.7 compares an integer, and how finely that integer is resolved decides how gently the unit can change. Measure the *collision probability* of the pre-activation, the chance that two random inputs give it the same value: that is the share of the unit's outputs that flip together when the folded threshold crosses one value. On random ternary weights at the measured zero fraction and $n = 1{,}152$, with no training involved, it is 0.0222 for the one-bit additive unit, 0.0101 with a second activation bit, 0.00063 with a rank-4 pair term at one bit, and 0.00011 with both. Equivalently the pre-activation takes about 11 distinct values per standard deviation in the first case and about 267 in the third, and the largest single value holds 3.2 percent of all inputs against 0.15 percent. So the one-bit additive unit cannot change its output by a little: the smallest move its threshold can make takes a large group of inputs with it. The reason is arithmetic rather than statistical. Both units emit an integer, so the smallest step either can take is 1; but a pair term multiplies two counts, each of order $\sqrt{n}$, so the integer it contributes is quadratically larger than the linear term's. The spread of the pre-activation grows about fiftyfold while the step stays 1, and the same step therefore cuts the distribution that much more finely. The scaling that follows is measured rather than asserted. On the untrained unit at ternary weights and the measured zero fraction, with $R = 4$ and the ratio of the two pre-activation spreads averaged over twenty weight draws, it runs 11.6, 15.1, 22.6, 29.2, 45.2 and 60.7 as n goes 72, 144, 288, 576, 1,152 and 2,304. Divided by $\sqrt{n}$ those are 1.37, 1.26, 1.33, 1.22, 1.33 and 1.27, flat to within an eighth over a thirty-twofold range of n. That is the scaling the abstract states. A single weight draw is not enough to see it: the draw-to-draw spread of the ratio is about a sixth of its value at every n measured. Read the other way, the additive unit's output depends on the input only through one signed count, so every input sharing that count collides; the pair term makes it depend on $2R$ further counts, which separates inputs the count alone cannot tell apart.

That ordering is the ordering of the measured instability. The four configurations move 0.85, 0.43, 0.26 and 0.21 percentage points per epoch late in training, ranking exactly as their collision probabilities do, with a correlation of 0.86 on the logarithm. Figure 2(c) shows the comb the threshold compares against and Figure 5(a) the staircase a drifting threshold therefore walks; the relation between the two is monotone across the four configurations. Both repairs raise the resolution, a product by about twenty-four-fold and a bit by about two, which is the order in which they steady training. We do not claim this is the only mechanism.

The same quantity inside a trained network. The numbers above are simulated on the unit with no training. Twelve instrumented runs, the four configurations at three seeds, record the collision probability per layer during real training and give the same answer. Averaged over the last ten epochs it is 0.01674 for the one-bit additive unit, 0.01312 with a second activation bit, 0.00070 with a rank-4 pair term at one bit and 0.00046 with both, ranking exactly as the simulated values do and exactly as the accuracy movement of those same runs, which is 1.181, 0.698, 0.329 and 0.270 points per epoch. The correlation on the logarithm is 0.906, against 0.913 for the simulated values. What a product buys is smaller in the network than on the unit, about twenty-four-fold against about thirty-five, and what a bit buys is smaller still, but neither ordering moves.

What carries the coupling. A rate cannot answer this: the mechanism names a share of outputs that flip *together*, and a per-layer rate records how much moved, not how much moved at once. Three rate-based readings were recorded in the same runs and none of them separates the configurations the way the mechanism needs. The coarser unit's emitted bits move *less* per epoch, 0.077 against 0.127 at one bit, while its accuracy moves three and a half times as much; that movement is no burstier, the largest epoch being 1.71 times the mean against 1.70; and the per-layer gradient norms do not separate degree at all, 0.696 against 0.762 in the first layer, the profile being set by activation precision instead.

A second set of instrumented runs records the quantity the mechanism actually names. For every output bit that flips between epochs it keeps the size of the group of inputs that shared that bit's pre-activation value, so a flip that takes a whole group with it is counted as one. Averaged over the last ten epochs and over seeds, that group is 0.03397 of a channel's inputs for the one-bit additive unit, 0.02221 with a second activation bit, 0.00121 with a rank-4 pair term at one bit and 0.00071 with both. The ordering is the ordering of the collision probabilities and of the accuracy movement, and the degree-1 to degree-2 ratio is 29.5, against 26.5 for collision. A threshold crossing in the additive unit does take about thirty times as large a group of inputs with it, which is what the resolution argument

Table 6: Collision probability by block at one activation bit, degree 1 over degree 2, from the instrumented runs of this appendix. The ratio is narrowest in the second block and never inverts.

block	1	2	3	4	5
degree 1 over degree 2	27.7	9.6	28.3	22.0	32.8

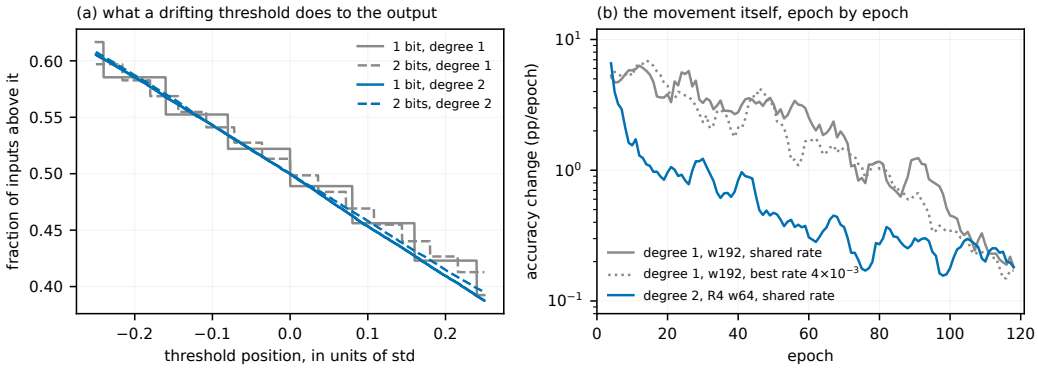

Figure 5: **What the coarseness of Figure 2(c) does, and the movement it explains. (a)** The fraction of inputs above the threshold as it drifts, on the untrained unit: the additive unit answers in steps, because each step crosses one of the few heavily loaded values, and the degree-2 unit smoothly. **(b)** The epoch-to-epoch change of the two curves of Figure 2(a) and of the baseline at its best matched rate, the quantity panel (a) accounts for.

claims. The relation is compressive rather than proportional, the group spanning forty-eight-fold across the four configurations where the movement spans under five, so we read the ordering and not a rate.

The same quantity along the depth of the network. These runs record every layer, and the gap between the repairs does not close anywhere in the stack (Table 6). The first block is the informative one: a second activation bit leaves the additive unit's resolution there unchanged, 0.06036 against 0.06032 with the seed bands overlapping, while degree changes it 27.7-fold; for the pair unit the bit gives 0.00218 against 0.00177, a real but small effect.

The reason is architectural. The first block reads the frozen thermometer code rather than an activation, so activation precision cannot alter its input at all and reaches it only through what the block learns once its own output quantizer changes, whereas a pair term acts on that input directly. This is the incomparability of Proposition D.1 showing up as a measurement, at a layer the proposition says nothing about, and it depends on the front end being frozen: a first block fed binarized activations should show the bit moving its resolution.

One reading we checked and discarded. The group, taken as a fraction of a layer's inputs, grows with depth for the additive unit, from 0.0080 in the second block to 0.0332 in the fifth, which invites the conclusion that it coarsens as it deepens. It does not. That fraction divided by the collision probability, which is what it would equal if flips fell at random, rises from about 1.4 in the first three blocks to about 5.5 in the last two *for all four configurations alike*, so the growth belongs to the layer geometry, the inputs per channel falling from 524,288 to 8,192, and not to degree. Counted in inputs rather than fractions the group shrinks with depth everywhere.

C Proofs

The proofs follow the order in which Section 3 states the results, and the two results stated in Appendix D are proved at the end.

Lemma 3.2. *Uniqueness of the multilinear expansion.* Substitute $x_i^2 = x_i$ in $x^\top Q x = \sum_i Q_{ii} x_i^2 + \sum_{i \neq j} Q_{ij} x_i x_j$ and collect (i, j) with (j, i). Two multilinear polynomials agreeing on $\{0, 1\}^n$ have the same coefficients, so g realizes f iff their pair coefficients agree; β and w absorb the constant and linear coefficients, the diagonal of Q included.

Lemma 3.3. *A dimension count, then an explicit block.* Lower bound: let $Q + Q^\top = S$ with $r = \operatorname{rank} Q < n_+(S)$; take a subspace V_+ of dimension $n_+(S)$ on which $v \mapsto v^\top S v$ is positive definite; since $\dim \ker Q = n - r > n - n_+(S)$, some $v \neq 0$ lies in $V_+ \cap \ker Q$, and then $v^\top S v = v^\top Q v + (Qv)^\top v = 0$, a contradiction. The splittings of $-S$ are exactly the $-Q$, so the same argument on $-S$ gives $\operatorname{rank} Q \geq n_-(S)$. Attainability: diagonalize $S = U \Lambda U^\top$; pair each positive eigenvalue λ with a distinct negative one μ while both remain, using the block $\left(\begin{smallmatrix} \lambda/2 & t \\ -t & \mu/2 \end{smallmatrix}\right)$ with $t = \sqrt{-\lambda\mu}/2$, which has $Q_2 + Q_2^\top = \operatorname{diag}(\lambda, \mu)$ and determinant $\lambda\mu/4 + t^2 = 0$, hence rank 1; each unpaired nonzero eigenvalue gets the 1×1 block $\lambda/2$ and each zero eigenvalue the zero block; assemble block-diagonally and conjugate by U. The rank is $\min(n_+, n_-) + |n_+ - n_-| = \operatorname{MaxIn}(S)$.

Theorem 3.4. By Lemma 3.2, realizability at rank R means a diagonal D and a Q with $Q + Q^\top = S_f + D$ and $\operatorname{rank} Q \leq R$ exist; for fixed D the minimal rank is $\operatorname{MaxIn}(S_f + D)$ by Lemma 3.3; minimize over D.

Corollary 3.5. *Eigenvalue shift, dimension count, explicit matrix.* (a) With eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ of S_f take $D = -\lambda_m I$, $m = \lceil n/2 \rceil$; $S_f + D$ has at most $m - 1$ positive and $n - m$ negative eigenvalues, so $\operatorname{MaxIn} \leq \lfloor n/2 \rfloor$. (b) Realizability at rank R gives $\operatorname{rank}(S_f + D) \leq 2R$ for some diagonal D. The variety of symmetric matrices of rank at most r has codimension $\binom{n-r+1}{2}$; the set of off-diagonal parts of rank-$\leq 2R$ matrices is the image of that variety under a linear map into a space of dimension $\binom{n}{2}$, so its Zariski closure has dimension at most $\binom{n+1}{2} - \binom{n-2R+1}{2}$, which is below $\binom{n}{2}$ as soon as $\binom{n-2R+1}{2} > n$, i.e. $n - 2R \geq q_n$; a proper subvariety is Lebesgue-null. (c) $Q = (c/2)\mathbf{1}\mathbf{1}^\top$ has rank 1 and off-diagonal c everywhere, so $\rho \leq 1$, and $\rho \geq 1$ since no diagonal D zeroes a nonzero off-diagonal; $\sum_{i<j} x_i x_j = \binom{s}{2}$.

ρ and the free-diagonal minimum rank. For symmetric M, $\operatorname{rank}(M) \geq \operatorname{MaxIn}(M) \geq \operatorname{rank}(M)/2$, so $\rho(f)$ lies between half and all of $\min_D \operatorname{rank}(S_f + D)$, the free-diagonal minimum rank studied in factor analysis; Corollary 3.5(b) is the inertia form of that literature's genericity threshold.

Proposition C.1 (BatchNorm fold)**.** *Let $y = g(x) - \beta \in \mathbb{Z}$ and $\mathrm{BN}(g) = \gamma(g - \mu)/\sigma + \beta_{\mathrm{BN}}$ with $\sigma > 0$. For $\gamma \neq 0$ put $\theta = \mu - \beta - \beta_{\mathrm{BN}}\sigma/\gamma$. For $\gamma > 0$, $\operatorname{act}(\mathrm{BN}(g)) = \mathbf{1}[y \geq \Theta]$ with $\Theta = \lfloor \theta \rfloor + 1$; for $\gamma < 0$, $\operatorname{act}(\mathrm{BN}(g)) = \mathbf{1}[-y \geq \Theta]$ with $\Theta = 1 - \lceil \theta \rceil$, i.e. the same form after negating every coefficient; for $\gamma = 0$ the unit is the constant $\mathbf{1}[\beta_{\mathrm{BN}} > 0]$.*

Proof. For $\gamma > 0$, $\mathrm{BN}(g) > 0 \iff g > \mu - \beta_{\mathrm{BN}}\sigma/\gamma \iff y > \theta \iff y \geq \lfloor \theta \rfloor + 1$ for integer y; $\gamma < 0$ reverses the inequality, $y < \theta \iff y \leq \lceil \theta \rceil - 1 \iff -y \geq 1 - \lceil \theta \rceil$; $\gamma = 0$ leaves the sign of β_{BN}.

Proposition 3.7. *The results above, plus the width hypothesis.* On $\{0, 1\}$, $x_i x_j = \mathrm{AND}(x_i, x_j)$, so Definition 3.6 evaluates equation 2, an identity on the cube; the coefficients are integers, so nothing is rounded; the threshold comparison is exact by Proposition C.1; and every partial sum lies in $[-M, M]$, which an ω-bit accumulator holds. Clamping Θ is free because the sum lies in $[-M, M]$: at $\Theta \leq -M$ the comparison holds on every input, giving the constant gate 1, and at $\Theta \geq M+1$ it holds on none, giving the constant gate 0, so every threshold outside $[-M, M+1]$ computes the same function as the endpoint it passes.

Crossover and quarter-square. At the charges of Section 3.5 the pair form costs $2(n + |P|) + |P|$ and the factored form $2(2R+1)n + 45R$, so the pair form is cheaper iff $|P| < (4Rn + 45R)/3$; at $n = 1{,}152$, $R = 4$ that is 6,204 pairs, 0.94 percent of $\binom{n}{2}$. Quarter-square: $(u+v)^2 - (u-v)^2 = 4uv$ identically over the integers, and with ternary a_r, b_r the vectors $a_r \pm b_r$ have entries in $\{-2, \ldots, 2\}$, so $u_r \pm v_r$ are weighted popcounts with $|u_r \pm v_r| \leq 2n$, each product is two lookups in a squaring table over $[0, 2n]$, and the comparison is carried out on $4y$ against 4Θ, exact in integers.

The two results stated in Appendix D. Proposition D.1 and Proposition D.2 are proved where they are stated; the two that close the threshold question are proved here, because both rest on Corollary 3.5.

Corollary D.3. *A count against Proposition D.2 (Appendix D).* By that proposition, $\log_2 |\mathcal{D}_R| \leq (2R{+}1)n^2{+}n$. The degree-2 threshold functions on the cube satisfy $\log_2 |T_2(n)| = n^3/2{\cdot}(1{+}o(1))$ (Baldi & Vershynin, 2019a, Remark 2.12), generalizing Baldi & Vershynin (2019b). Since $\mathcal{D}_R \subseteq T_2(n)$ on the same 2^n points, the realizable fraction is at most $2^{(2R+1)n^2+n-(n^3/2)(1-o(1))}$. At $R = o(n)$ the positive terms are $o(n^3)$, so the exponent is $-(n^3/2)(1-o(1))$ and the fraction tends to 0. Conversely a non-vanishing fraction forces $(2R{+}1)n^2 + n \geq n^3/2 \cdot (1-o(1))$, hence $R \geq (1/4 - o(1))n$.

Proposition D.4. *One explicit family; Corollary 3.5(c), then (b).* Take $c_{ij} = 2 + \varepsilon g_{ij}$ with g symmetric, $|g_{ij}| \leq 1$, zero diagonal and generic, and $0 < \varepsilon < 1/n^2$. Then $f(x) = 2\binom{s}{2} + \varepsilon \sum_{i<j} g_{ij} x_i x_j$ with s the input popcount. First half: the perturbation is below $1/2$ in absolute value because $\binom{n}{2}\varepsilon < 1/2$, and $2\binom{s}{2}$ is an even integer, so at a half-integer threshold $\theta = 2t - 1$ we have $f(x) \geq \theta$ iff $\binom{s}{2} \geq t$. That unit is $\mathbf{1}[\binom{s}{2} \geq t]$, which Corollary 3.5(c) realizes at rank 1. Second half: $S_f = 2(\mathbf{1}\mathbf{1}^\top - I) + \varepsilon g$ is an invertible affine image of g, so it carries full measure to full measure; a generic g therefore places S_f outside the null set of Corollary 3.5(b), and $\rho(f) > (n - q_n)/2$. Computed at $n = 8, 10, 12$: the Boolean function agrees with the popcount unit on every input, while $\rho(f)$ is 4, 5 and 6 against rank 1.

D The two repairs are incomparable

Fix a layer with $n \geq 4$ integer pre-activations $z = (z_1, \ldots, z_n)$, each z_i ranging over a finite set Z_i of integers, the admissible inputs being the product of the Z_i; a unit of the next layer reads z through a per-channel quantizer and returns one bit. With $\phi_i : \mathbb{R} \to \{0, \ldots, 2^b - 1\}$ monotone step functions, $\mathcal{L}_b(n)$ (the precision route) is the set of $g(z) = \mathbf{1}[\sum_i w_i \phi_i(z_i) \geq \tau]$ with $w \in \mathbb{R}^n$ and free quantizer thresholds; $\mathcal{D}_R(n)$ (the degree route) fixes the one-bit code $x(z) = (\mathbf{1}[z_i \geq t_i])_i$ and is the set of $g(z) = h(x(z))$ with h a rank-R degree-2 threshold function of equation 1; $\mathcal{D}_0(n)$ has no pair term. Because the code is fixed, each $g \in \mathcal{D}_R(n)$ is determined by its h, so in the counting arguments below we identify $\mathcal{D}_R$ with the corresponding set of Boolean functions on $\{0,1\}^n$; that is the set the sign-pattern bound counts, and the set Corollary D.3 compares against $T_2(n)$.

Proposition D.1 (incomparability)**.** *(i) $\mathcal{D}_1(n) \not\subseteq \mathcal{L}_b(n)$ for any $b \geq 1$; witness x_1 XOR x_2, whenever both code bits are non-constant on the admissible inputs. (ii) $\mathcal{L}_b(n) \not\subseteq \mathcal{D}_R(n)$ for any $b \geq 1$, $R \geq 0$ and fixed code; witness a second threshold on one channel, $\mathbf{1}[z_1 \geq t']$, whenever t' and t_1 induce different bipartitions of Z_1, each with two nonempty parts.*

In words: $\mathcal{L}_b(n)$ is what the precision repair can compute, a unit that reads each channel through its own quantizer and then adds; $\mathcal{D}_R(n)$ is what the degree repair can compute, a unit that reads every channel through one fixed one-bit code and may multiply the code bits in pairs. Part (i) says that adding activation bits never reaches the exclusive-or of two channels. An additive unit's response to one channel does not depend on the other, so whatever two values channel 1 contributes and whatever two channel 2 contributes, the four sums pair up and cannot alternate in sign the way exclusive-or requires; more bits refine what each channel contributes and leave that pairing untouched. Part (ii) says that products never recover what the code discarded. Every function in $\mathcal{D}_R$ is constant on inputs sharing a code, so two pre-activations on the same side of t_1 but opposite sides of a second cut t' must receive the same output, however many products the unit has, while a two-bit quantizer reaches the second cut by placing a threshold at t'. So neither class contains the other, and neither witness is exotic: an exclusive-or of two channels, and a second threshold on one channel's magnitude.

Proof; (i) additivity across channels, (ii) the fixed code. (i) $x_1 \,\text{XOR}\, x_2 = x_1 + x_2 - 2x_1x_2$ and $x_1x_2 = (e_1^\top x)(e_2^\top x)$, so the function is in $\mathcal{D}_1$. If it were in $\mathcal{L}_b$, fix the other channels, fix admissible inputs $\alpha_0 < t_1 \leq \alpha_1$ and $\beta_0 < t_2 \leq \beta_1$, and set $U_k = w_1\phi_1(\alpha_k)$, $V_k = w_2\phi_2(\beta_k)$; XOR needs $U_0 + V_0 < \tau$, $U_1 + V_1 < \tau$, $U_0 + V_1 \geq \tau$, $U_1 + V_0 \geq \tau$, and adding the first pair, then the second pair, gives the same sum on both sides of a strict inequality. (ii) Every function in $\mathcal{D}_R$ depends on z only through the fixed code $x(z)$, hence is constant on each set of inputs sharing a code; let

$A = \{v \in Z_1 : v \geq t_1\}$ and $B = \{v \in Z_1 : v \geq t'\}$, nested and distinct; if $B \subsetneq A$ take $v \in A \setminus B$ and $v' \in B$, otherwise $v \in B \setminus A$ and $v' \in Z_1 \setminus B$, nonempty by hypothesis; in both cases v, v' share a code bit and differ in g; and $g \in \mathcal{L}_b$ for every $b \geq 1$ by placing one free threshold at t'.

The capacity upper bounds below do not order the two classes either.

Proposition D.2 (capacity bounds)**.** *In the capacity of Baldi & Vershynin (2019a), the binary logarithm of the number of realizable functions on a finite input set, the fixed-quantizer subclass $\mathcal{L}_b^{\text{fix}}$ satisfies $C(\mathcal{L}_b^{\text{fix}}) \leq bn^2$ and $C(\mathcal{D}_R) \leq ((2R{+}1)n{+}1)\,n$ for $n \geq 4$, while the one-bit linear unit has capacity $n^2(1 + o(1))$; the free-threshold class is not bounded here. In the fixed-quantizer subclass a second activation bit thus at most doubles the linear unit's capacity and a rank-R term raises it by a factor of at most $2R{+}1$, up to lower-order terms.*

Proof; set counting, Warren's bound, Zuev's theorem. For $\mathcal{L}_b^{\text{fix}}$ each g is a threshold function of $(\phi_i(z_i))_i$ on a set of 2^{bn} points, and for $n \geq 4$ the number of threshold functions on a finite set $S \subset \mathbb{R}^n$ is at most $2^{n \log_2 |S|}$ (Baldi & Vershynin, 2019a, Lemma 2.7). For $\mathcal{D}_R$, h is the sign of a polynomial of degree 2 in $p = (2R{+}1)n + 1$ parameters (including the bias) evaluated at $m = 2^n$ points; if $p > m$ the trivial bound $C(\mathcal{D}_R) \leq m < pn$ applies; otherwise the number of sign patterns of m polynomials of degree at most d in $p \leq m$ variables is at most $2(2emd/p)^p$ (Warren's bound in the form of Bartlett et al. (2019, Lemma 17)), so $C(\mathcal{D}_R) \leq 1 + p(n + \log_2(4e/p)) \leq pn$ for $p \geq 12$. The only case left is $p < 12$ with $n \geq 4$, which forces $R = 0$ and $n \leq 10$; there $\mathcal{D}_0$ is a threshold function on the 2^n cube points and Lemma 2.7 gives $C(\mathcal{D}_0) \leq n^2 \leq pn$. The linear unit's capacity $n^2(1 + o(1))$ is Zuev's theorem (Baldi & Vershynin, 2019a, Theorem 2.11).

Does the threshold rescue the unit? The second scope clause is the question that matters in hardware; the next two close it from opposite sides.

Corollary D.3 (almost all degree-2 threshold functions need linear rank)**.** *Let $T_2(n)$ be the Boolean functions on $\{0,1\}^n$ of the form $\mathbf{1}[f \geq \theta]$ with f multilinear of degree at most 2. If $R = o(n)$, the fraction of $T_2(n)$ computed by rank-R units tends to 0; a non-vanishing fraction forces $R \geq (1/4 - o(1))\,n$.*

Proposition D.4 (thresholding collapses the rank a function needs)**.** *There are degree-2 polynomials f with $\rho(f) > (n - q_n)/2$ whose threshold $\mathbf{1}[f \geq \theta]$ a rank-1 unit computes.*

Counting says the sign buys nothing in general (Corollary D.3); one function says it can buy everything (Proposition D.4), a perturbed all-pairs term whose threshold a one-product popcount unit still computes.

E Numerical verification of the capacity results

The statements of Section 3.2 are proved in Appendix C. This appendix reports an independent check of the ones a machine can confirm, so that a reader can separate what the proofs assert from what a program agrees with. The script is released with the code.

Two of the statements are constructions and are checked by carrying the construction out and measuring the residual. One is a bound and is checked by exhibiting the diagonal that attains it. The remaining two are statements about Boolean functions on few inputs and are checked by enumerating every input. What is deliberately absent is a search for ρ. Since $\rho(f)$ is a minimum over diagonals, a search returns upper bounds only, and a search that fails to find a diagonal is not evidence that none exists; no number below comes from one.

Table 7 collects the outcomes. Three points are worth drawing out.

The free diagonal is what the rank-capacity theorem buys, and its effect is large rather than marginal. For the all-ones pair term the matrix with its diagonal held at zero has $n{-}1$ eigenvalues of one sign, so a unit that must reproduce it entrywise spends $n{-}1$ products; Corollary 3.5(c) says one suffices, because on the cube the diagonal is free and a single shift makes the matrix rank one. The saving is exact, not an approximation.

The two directions of Theorem 3.4 are not equally checkable, and we state which is which. Sufficiency is constructive: given a diagonal attaining the minimum, the splitting of Lemma 3.3 produces the factors, and the check above confirms the result is exact. Necessity is proved and cannot

Table 7: Numerical check of the capacity results. Every entry is an exact construction or an enumeration of all inputs, at double precision; no entry comes from a search over diagonals. Residual is the largest absolute entrywise discrepancy over all draws.

Statement	What is checked	Range	Outcome
Lemma 3.3	$Q + Q^\top = S$ and rank $Q = \mathrm{MaxIn}(S)$	$n = 6$ to 32, 40 draws each (half of them rank-deficient)	holds in every draw residual below 10^{-13}
Cor. 3.5(a)	a diagonal with MaxIn $\leq \lfloor n/2 \rfloor$	$n = 6$ to 32, 40 draws each	attained in every draw
Thm. 3.4, Cor. 3.5(c)	rank with the diagonal held at zero, against rank with it free	$n = 6$ to 16	$n-1$ against 1
	the rank-1 unit against f	all 2^n inputs	residual below 10^{-13}
Prop. D.4	thresholded perturbation against the rank-1 popcount unit	$n = 8$ to 14, all 2^n inputs perturbation below 10^{-2}	identical at every input margin to threshold $1/2$
Prop. D.1(i)	an additive unit against exclusive-or	one to four activation bits, all input pairs	no additive unit; one product suffices

be confirmed by computation on an instance, because certifying that no diagonal does better is the minimum-inertia problem itself.

Proposition D.4 has one half that a machine settles, the Boolean identity in the table, and one it does not: that ρ of the underlying polynomial is linear in n follows from the genericity argument of Corollary 3.5(b), so the values quoted in Appendix C for $n = 8, 10, 12$ come from that argument and not from fitting a unit.

What the pair term adds, counted exactly. The rank condition says which degree-2 functions a unit of a given rank realizes. At small n the prior question, how much the pair term buys at all, is settled by enumeration. For every Boolean function on n inputs we test realizability by one unit as a linear program in the unit's coefficients, at degree 1 and at degree 2. A degree-1 unit realizes 14 of the 16 functions at $n = 2$, 104 of 256 at $n = 3$ and 1,882 of 65,536 at $n = 4$; a degree-2 unit realizes all 16, then 254, then 57,574. At four inputs the pair term multiplies the reachable set about thirtyfold, and the exclusive-or of two channels is one gate among the 55,692 gained. Three of the gained gates say what kind of computation this is: the exclusive-or itself; *exactly two of three inputs are on*, which is a band on the count rather than a threshold on it; and the multiplexer *if* x_3 *then* x_1 *else* x_2, one channel selecting which linear function of the others applies, which is conditional computation inside a single unit. The boundary is as sharp: parity of three inputs and of four stays out of reach at degree 2, and the two functions of 256 that a degree-2 unit misses at $n = 3$ are exactly parity and its complement. The degree-1 column is the classical count of threshold functions, which the enumeration reproduces.

F Deployment check

What hardware-aligned means, element by element. Section 1 calls a model hardware-aligned when its deployed form is a circuit of integer adders, comparators, logic gates and a bounded number of integer multipliers per unit, with no floating point, and its deployed function is the one that was trained. The four element types are the vocabulary of digital circuit design (Harris & Harris, 2012; Weste & Harris, 2010), and for a reader who does not work with hardware they are, in order of cost: a *logic gate* takes one or two bits and returns one bit (AND, OR, XNOR), and is the cheapest element, one lookup-table cell on an FPGA; an *integer adder* takes two integers of w bits and returns their sum, at a cost that grows linearly with w, and an adder tree sums n such integers in $\lceil \log_2 n \rceil$ levels; a *comparator* returns one bit saying whether one w-bit integer is at least another, again at a cost linear in w; an *integer multiplier* takes two w-bit integers and returns their product, at a cost that grows with w^2, which makes it the expensive element and the one whose count a designer bounds. A *floating-point* unit adds an exponent, a mantissa and normalization logic around each of these and costs many times its integer counterpart at the same width (Horowitz, 2014), which is why a circuit target has none. With these meanings, each clause of the definition names one thing in the compiled unit of Section 3.3: the weighted counts are the adder trees, the output bit is the comparison against Θ, the AND of two data bits in the pair form is the only runtime logic gate, and the R products of

the factored form are the bounded multiplier count. That bound is what separates the unit from a multiply-accumulate array, which needs a multiplier per input per unit, the cost binarization removed and this paper does not put back. Two clauses do work that is easy to miss. *No floating point* excludes the readout from the body's claim, which is why an integer head is measured alongside the float one (Section 4.2). *The deployed function is the trained one* excludes post-training quantization, which converts a float model after the fact, and a relaxed unit snapped to a table after training, which fails whenever the snap changes the function. Any repair that keeps a real-valued path between layers, a float shortcut or a float scale that does not fold into Θ, fails the first clause, which is why the precision repair considered here is the one that quantizes the activation itself.

The readout and fixed-pair controls in full. The deployed body has no floating-point element once the head and the readout are integer as well. The count readout scores a class by summing a fixed group of channels. It therefore needs channels to spare per class and is defined here only for the ten-class datasets; CIFAR-100 at these widths would give two channels per class on the degree-2 model and would measure the readout rather than the unit. It is not run. All four combinations of a float or ternary head with a linear or count readout were trained together in one directory under the shared recipe, three seeds each, leaving no cell spliced across recipes; the float-head, linear-readout corner returns 84.78 ± 0.08 and 86.96 ± 0.08 against Table 1's 84.84 and 86.96, which is the check that it is the same setup. The fully integer corner gives $+1.61$ pp (Table 3). Read against the default, each of the three changes lifts the degree-1 model, by 0.50, 0.78 and 0.52, and moves the degree-2 model by -0.08, -0.05 and -0.06. Every edge in the grid narrows because the baseline improves, not because the pair unit loses. We claim no mechanism for that, and it is not general. On the three datasets with both models complete the degree-1 model moves $+0.51$, $+0.07$ and $+0.54$ on CIFAR-10, Fashion-MNIST and KMNIST while the degree-2 model moves -0.05, $+0.02$ and $+0.06$: the baseline lift appears on two of the three and the pair unit is unmoved on all of them. WHAT SURVIVES EVERYWHERE is the edge itself, $+1.61$, $+1.42$ and $+1.11$ with disjoint seed bands on every one, against $+2.18$, $+1.46$ and $+1.59$ at the float head. The deployed network therefore keeps the degree gain without a floating-point element anywhere, and where the edge narrows it is because the baseline improves. CIFAR-100 is the exception on both counts. The count readout cannot run there, leaving the ternary head with a linear readout as what deploys, the last row of Table 3; that head moves the degree-1 model $+0.35$ and the degree-2 model -1.09, so the edge falls from $+2.05$ to $+0.62$ pp, bands still disjoint. It is the only pair measured where the head costs the pair unit rather than lifting the baseline: the same change moves the degree-2 model -0.05 on CIFAR-10, -0.09 on Fashion-MNIST and -0.11 on KMNIST, where it lifts the degree-1 model $+0.78$, $+0.44$ and $+0.71$. It is the hundred-class problem that is unlike the others, not the configuration. Taking the float out of the head is therefore not free on every dataset, and on the hundred-class problem it is the pair unit that pays. The $\rho = 1$ structure of Corollary 3.5(c) gives a one-bit control: fixing the pair matrix to all ones and learning one scalar per output channel, width held, returns $+0.05$ pp with overlapping seed bands where the learned unit returns $+2.11$, so the gain needs learned pairs and not the extra convolution (Table 3). On real activations the factored and pair forms agree exactly on every layer (below).

Three competing explanations of the gap. The degree-2 models carry 1.47 and 2.39 percent more parameters, worth 0.02 to 0.04 pp where a doubling is worth $+1.02$ pp, and the deficit is measured directly rather than extrapolated: `A1d2R4w63` (Table 2), at 4,949,227 parameters or 1.64 percent *below* the baseline and at $0.997\times$ its logic, so behind it on both currencies at once, scores 86.84 ± 0.24 against its 84.84 ± 0.04, an edge of $+2.00$ with disjoint seed bands, so moving 3.1 percent of the parameters across the comparison moves the edge 0.11. That cell was run on every dataset of Table 1, at a deficit of 1.64 percent on CIFAR-10 and SVHN, 2.52 on CIFAR-100 and 2.72 on Fashion-MNIST and KMNIST: the edges are $+2.00$, $+1.78$, $+2.07$, $+1.57$ and $+1.57$ pp against matched-parameter edges of $+2.11$, $+1.71$, $+2.05$, $+1.46$ and $+1.59$, every seed band disjoint and three of the five larger at the deficit than at parity. Taking parameters away from the pair unit does not take the gain with them, on any of the five. The degree-1 baseline reads up to three times as many inputs per unit, not fewer; and removing the pair term's diagonal at inference changes nothing once BatchNorm is refit (below).

Per-layer identity. The factored training form and the pair form are algebraically identical (Proposition 3.7). Evaluated per layer on real activations of the rank-1 1.7M anchor (40-epoch recipe), three

seeds, sixteen images, they agree to a maximum difference of 0 in float32 on every layer, exact because every operand is an integer below 2^{24}. The check is on the pre-BatchNorm accumulator; the fold is a proposition, and we claim no measured end-to-end integer equivalence beyond it.

Scaling the fixed all-ones control. A first attempt at that control, at 40 epochs, diverged: it diverged at 24.97 percent, the fixed pair term being 233 to 1,382 times the learned scale; the model reported divides by n, a reparameterization of the free scalar. The model reported in Table 3 uses that reparameterization.

Pruning the pair form. One post-hoc magnitude-pruning test, no retraining: a trained unit at a fan-in-72 anchor, 40 epochs, one seed, BatchNorm refit. Keeping half the pairs, rank 1 falls $69.99 \rightarrow 61.82$ and rank 4 falls $76.84 \rightarrow 73.80$; keeping a quarter, 52.98 and 63.33. The crossover of Section 3.5 at that fan-in is 4.3 percent of the pairs at rank 1 and 17 percent at rank 4, so half the pairs is 12 and 3 times that density: accuracy is already gone far above the crossover. Pair-form sparsity trained in is not tested.

Post-hoc diagonal removal. The pair term's diagonal lands on the linear coefficients, so a degree-2 unit compiles to a wider integer alphabet than a ternary one: in `A1d2R4w64` at seed 0, 27 percent of those coefficients are zero and 43 percent of the nonzero ones lie beyond ± 1. Remove that diagonal from the final `A1d2R4w64` checkpoint at seed 0, which scores 86.95 as trained against that seed's last-10 accuracy of 86.98. Accuracy falls to 74.61 with no BatchNorm refit and returns to 86.82 with a refit over 200 batches of 256 training images, labels unused, against 86.86 for the same refit of the unmodified network. The refit absorbs a per-channel statistics shift, on one draw. This is an inference-time test; whether training needs the wider alphabet is the first control below.

Where activation bits enter the cost. Activation bits enter a circuit's cost in three places, and each is where a one-bit activation is the cheaper choice. Stored bits per image grow linearly with b, so once a layer's map no longer fits on-chip every added bit is off-chip traffic; every tap of every adder tree widens with b, three, four and five full-adder cells at one, two and three bits in the model of Section 3.5, and the tree is over 97 percent of a unit's logic; and in a streaming design each activation bit occupies a signal line and a register stage between layers, so routing and fan-out scale with b as well.

What each count sees at the matched point. Taking the matched-parameter pair of Table 1, degree 2 over degree 1, and reading every currency the quantization literature uses: parameters 1.015, training MACs 1.000, bit operations 1.000, deployed logic 1.028, stored activation bits 0.333. The first four are counts of WORK and call the two the same model; only the memory count separates them, and it separates them by a factor of three. That is the paper's comparison in one line, and it is why an accuracy-only or a MAC-only reading of these two models finds nothing to choose between them.

The two cost counts in full. Every network in this paper carries two cost measures. The first is logic, counted in LUT-equivalents (a LUT is a lookup-table cell, the measure of area on an FPGA) by an analytical model of the body, the convolutional layers between the input encoding and the readout, the final linear classifier (Section 3.5): one LUT-equivalent is the charge for a 2-input gate, and Section 3.5 gives the rest. The weight-shared view counts one circuit per output channel, reused across positions; the unrolled view counts one per position; we give both wherever they differ by more than ten percent. The count is not a synthesis result: it charges every adder tap, including taps a zero weight deletes in hardware, and the readout is excluded on every network, which favors the wider degree-1 head. The second cost measure is stored activation bits: what a layer writes for the next to read, summed over the body per image. A layer-sequential accelerator holds one layer's activation map between passes and a streaming one such as FINN holds a sliding window of it, so this count is the traffic one layer hands the next, not the size of any one memory. It is not the parameter count. Parameters are weights, stored once, growing with the branch count and the square of the width; stored bits are written once per image and grow only with the width and the bits per activation. So networks matched on parameters need not be matched on stored bits: a degree-2 network pays for its branches by being narrower, and narrower is fewer activations to store. Memory and routing lie outside a logic model, so the two counts sit beside each other and are never added.

What carries over at two and three bits. Three things carry over and two do not. The factored form carries over: it is still $2R{+}1$ weighted sums, now of b-bit values, and R integer multiplies of those sums, which is how Section 3.5 charges it. The threshold fold carries over: in the integer levels k, scaling both sides by $(2^b-1)^2$ leaves the pair term as it is and multiplies the linear term by 2^b-1, so each of the 2^b-1 quantizer thresholds of the next layer folds to a comparison between two integers. And the parameter and logic matching of Section 3.1 carries over, since it depends on the branch count and the width, not on b. What does not carry over is Lemma 3.2, since $x_i^2 = x_i$ fails off the cube, so there is no pair form and no AND gate at $b \geq 2$, only the factored form; and with it the rank theory of Section 3.2, which characterizes exact realization on the cube and is not restated for wider inputs. One caveat is specific to the three-bit models: their quantizer learns its range per layer, which places one real scale per layer in the linear coefficients after the fold; compiling it exactly needs a fixed-point constant per layer, which this paper does not verify; the two-bit models use a fixed clamp and fold as above. The per-layer identity check above was run at one bit. In every other respect the circuit at b bits is the circuit at one bit: ternary weights still compile to wiring, a $+1$ routing an activation's b bits into the positive count, the R multiplies are the only multipliers, and nothing is float. What b adds is adder width and activation memory, which is what Section 4.3 compares against the pair term.

The cost model, element by element. The cost model charges four elements: an adder tap, a multiply, a b-bit quantizer and an AND. Three of the four charges are assumed, not derived. An adder tap costs two LUT-equivalents, the assumed cost of its three full-adder cells, scaled as the tree widens (tending to four and five cells at two and three bits). A multiply costs 45 at one bit, at every fan-in. Its operands are signed popcounts about $b + \log_2 n$ bits wide, and the charge grows with the square of that width against its one-bit value. The precision penalty therefore shrinks as fan-in grows: 63 at three bits at $n = 1{,}152$, and 68 at $n = 216$. A b-bit quantizer costs $2^b - 2$ per unit. Only the AND charge, one LUT-equivalent, is definitional. Table 2 charges the widened costs; the crossover below uses one-bit counts. The pair form is cheaper only while $|P| < (4Rn + 45R)/3 \approx \frac{4}{3}Rn$, at $R = 4$ under one percent of the pairs at $n = 1{,}152$ and about 5 percent at $n = 216$. Trained units are far denser, and pruning them that far costs accuracy (below), so we deploy the factored form and keep the pair form as the exact reference. No comparison turns on the assumed multiply charge: the tree is at least 97.5 percent of the LUT-equivalent cost on every one-bit degree-2 network on either view, and the 1,600 multiplies of `A1d2R4w40` (names are decoded in Section 4.1) are 1.75 percent of its cost; The next paragraph recomputes both charges.

Recomputing the cost model. At three times the assumed multiply weight, `A1d2R4w40` would carry about 5 percent more logic than `A1d1R0w122`, worth under 0.1 pp on the width curve. Even at 300 per multiply with taps charged a third as much, `A1d2R4w40` carries 1.35 times `A1d1R0w122`'s logic and the matched-logic gap stays near 1.8 pp. On the degree axis matched logic is therefore close to matched parameters, since the adder tree dominates. The two counts separate on the precision axis, where `A1d1R0w110` and `A2d1R0w110` hold equal parameters at $0.330\times$ and $0.441\times$ the logic.

Why the diagonal control is not informative. A degree-2 unit *trained* with its pair-term diagonal removed, so that its compiled linear coefficients are exactly ternary, would not isolate what it appears to: it is unmatched on one extra convolution per layer, on the removed term's gradients, and on linear tap density, so a difference could not be attributed to the diagonal. The post-hoc removal above is the clean version of that test.

The pair term at two bits and the shorter budget. At the shorter 40-epoch budget of Appendix H, whose recipe differs from the shared one, the two-bit degree-1 model at width 192 scores 86.84 ± 0.16 against 86.48 ± 0.07 for the two-bit degree-2 model at matched parameters: the pair term trails by 0.36 pp. It is the one cell in this paper where the pair term is behind at matched parameters, and it is reported here rather than left to the tables.

Does the buffer bind? The exchange is stated in stored bits because that is what one layer hands the next, whether an accelerator buffers a whole map or streams a window of it, and the exchange matters only where it binds. On the models measured here it does not: the largest per-image total is 234 KiB, small enough to sit in on-chip memory on a mid-range part, and a layer-sequential design holds one layer's map at a time, less again. The regime where it binds is therefore one this paper

Table 8: The learning-rate sweep. Test accuracy on CIFAR-10, 3 seeds, everything but the rate held at the shared recipe; 10^{-3} is the shared learning rate that every number in Section 4 uses, called the shared rate below. A dash is a configuration that was not run. The three rates at which both degrees were run are the matched grid; the baseline was also run at 5×10^{-4}, below the shared rate. The last row is the baseline with its float head replaced by a ternary one, at the two rates it was run at; it is a degree-1 row and enters no comparison across degrees.

model	5×10^{-4}	10^{-3}	2×10^{-3}	4×10^{-3}
`A1d1R0w192`	83.74 ± 0.04	84.84 ± 0.04	85.77 ± 0.18	86.67 ± 0.30
`A1d2R4w64`	–	86.96 ± 0.07	87.05 ± 0.31	86.84 ± 0.12
`A2d1R0w110`	–	86.54 ± 0.14	87.31 ± 0.05	87.77 ± 0.16
`A1d1R0w192`, ternary head	–	85.56 ± 0.07	–	86.56 ± 0.20

does not measure, and the logic side is modeled rather than synthesized, so neither cost is settled on a device. What the measurements support is the relative statement at a held budget, not a claim about a deployed device.

Which side the binarization cost is on. The 4.72 pp sits almost entirely on the activation axis. At width 192 under the shared recipe, float weights with float activations score 89.56 ± 0.10 and ternary weights with float activations 89.60 ± 0.43: ternarizing the weights costs nothing at this spread. Float weights with one-bit activations score 84.69 ± 0.21 against 84.84 ± 0.04 for the ternary pair, putting the cost of binarizing the activations at 4.87 pp. The two single-axis costs sum to 4.82 against a joint 4.72, an interaction of 0.10, so the axes are additive to within the seed bands. At one-bit activations the weight alphabet does not separate the models at all: binary 84.95 ± 0.18, ternary 84.84 ± 0.04 and float 84.69 ± 0.21 span 0.26 pp across a 256-fold change in weight precision. That is why this paper repairs the activation side. The same split shows up in the binarization literature on a different network and dataset: Rastegari et al. (2016, Table 1) report full-precision AlexNet at 56.6 percent top-1 on ImageNet, 56.8 with binary weights, and 44.2 once the inputs are binarized as well. There too the weight side is free and the activation side carries the loss. Two caveats ship with the split: both one-bit models are still gaining at this budget, making the 4.87 a budget-limited estimate that more epochs would reduce; and the two float-activation cells keep a float element, which makes them attribution controls rather than models we propose.

Distance to the float reference. With the integer head of Section 4.2 the network deploys with no floating-point operation anywhere; the first layer already reads binary planes. `A1d2R4w64` sits 2.61 pp below the like-for-like float network with the float head and 2.73 pp with the integer head, a design-point distance; the ternarization inside it costs nothing, as the next paragraph measures.

G Learning rate, convergence, and what tuning changes

This appendix documents the one instability in this paper. Every number in Section 4 uses the codebase default of 10^{-3}, fixed before any degree-2 run, with AdamW, weight decay 5×10^{-4}, batch 128 and a cosine schedule over 120 epochs. That rate is not each model's best, and the two degrees are not equally sensitive to it. Three seeds per configuration. Both degrees were run at 10^{-3}, 2×10^{-3} and 4×10^{-3}, and every comparison across rates below is taken over those three: a rate at which only one degree was run is not a comparison, and is not reported as one. Table 8 gives the sweep.

The two curves have different shapes. The baseline gains about a point per doubling of the rate and has not turned over inside the grid: at 4×10^{-3} it still gains 4.28 pp over its last 60 epochs. The degree-2 model gains 0.10 pp from the shared rate to 2×10^{-3} and then falls 0.21 pp, so it peaks at 2×10^{-3}. At each model's best rate on the matched grid the pair unit leads by 0.38 pp, against 2.11 pp at the shared rate, and the seed bands overlap: tuning the baseline closes the gap to inside seed noise, and the paper claims no accuracy gap over a tuned baseline.

Instability along iteration. Figure 5(b) measures the same thing epoch by epoch. The baseline's test accuracy moves three times as far per epoch as the pair unit's over the first 60 epochs, 4.02 pp

Table 9: Which models have finished training at the 120-epoch budget. Residual is what a model still gains over the last 60 of its 120 epochs, the last-10 mean minus the epoch-51-to-60 mean; the schedule is shared, so the annealing component is shared. The float reference's residual is 1.70 at this budget. The upper two columns are the 120-epoch budget every table in this paper reports, split at the gap between 2.57 and 3.04. The lower block applies the same measure to the 240-epoch runs, over the last 60 of their 240; every one of those is inside the converged band, which is the point of running them.

converged	residual	still gaining	residual
`A2d2R4w64` (d2, 2 bit)	1.43	`A2d1R0w20` (d1, 2 bit, width 20)	4.31
`A2d2R4w36` (d2, 2 bit)	1.52	`A1d1R0w192` (d1, 1 bit)	5.56
`A3d2R4w64` (d2, 3 bit)	1.43	the ternary-head degree-1 model	5.33
`A1d3R4w42`	1.72	the tuned ternary-head degree-1 model	5.36
`A1d2R4w64` (d2, 1 bit)	1.89	`A1d1R0w110` (d1, 1 bit)	5.96
`A1d2R4w40` (d2, 1 bit)	2.14	`A1d1R0w122` (d1, 1 bit)	6.15
`A2d1R0w110` (d1, 2 bit)	2.41	the all-ones pair model	6.28
`A1d2R1w64` (d2, 1 bit)	2.57	`A3d1R0w192` (d1, 3 bit)	3.04
At 240 epochs: the four models of the two matched budgets, and the float reference. All five are converged.			
`A1d1R0w192` (d1, 1 bit)	1.43	`A1d1R0w122` (d1, 1 bit)	1.82
`A1d2R4w64` (d2, 1 bit)	0.44	`A1d2R4w40` (d2, 1 bit)	0.58
the float reference	0.43		

against 1.32, and still three times as far over the last 60, 0.85 against 0.26, with single-epoch swings of up to 2.71 pp in its last 30 epochs where the pair unit's largest is 0.61. At 4×10^{-3} the baseline is higher but not steadier: 0.69 per epoch over its last 60 and a largest late swing of 1.06 pp. The two degree-2 models are the steadiest curves in this paper at both rates measured. This is the instability the main text refers to, and it is why the shared recipe is the one at which the models are compared: the baseline's number is the one that depends on which rate and which epoch it is read at.

The instability belongs to the one-bit additive unit, not to width and not to the baseline's particular rate. Over the same window, every one-bit degree-1 model in this paper moves 0.66 to 1.02 pp per epoch (widths 110, 122 and 192, the integer-head model, and every rate of the sweep), and so does the all-ones pair model at 0.93, whose pair structure is fixed. Every degree-2 model moves 0.25 to 0.35 pp, at rank 1 as at rank 4 and at widths 40 to 96, as does the degree-3 model. Either repair removes it: a second activation bit takes the degree-1 model to 0.43 and a third to 0.27, the level of the float reference (0.18 to 0.24), and one learned product does the same at one bit. A fixed product does not. That one-bit training with the straight-through estimator is delicate is known: Alizadeh et al. (2019) report that the convergence of binary models relies on careful fine-tuning and that second-moment learning-rate adaptation is crucial for the estimator, which our use of AdamW follows. The observation our models add is that a graded term in the unit, whether an added bit or a learned product, is what steadies training, and the paragraph below gives a mechanism for it.

The reason is in the training curves rather than the test scores. At the shared rate the degree-2 model reaches 99 percent of its final accuracy at epoch 66 to 77 and gains 0.4 to 0.6 pp over its last 30 epochs; the degree-1 baseline reaches that point at epoch 81 to 87 and is still gaining 1.8 to 2.0 pp. Training accuracy agrees: 98.35 percent for the baseline at 2×10^{-3} against 99.06 for the degree-2 model at the same rate, rising to 99.30 when the baseline is given 4×10^{-3}. At each model's best rate the ordering of training fit therefore reverses, 99.30 against 99.06, while the test ordering does not, so at that point the runs no longer separate capacity from optimization. The baseline is budget-limited at the shared rate, and a larger rate buys it much of what a longer schedule would.

Which comparisons are between converged models. The diagnostic is what each model still gains over the last 60 of its 120 epochs, which Table 9 gives for every model in this paper.

The two groups are separated by a clear gap, 2.57 against 4.23, with nothing between. The four replication datasets were measured the same way: on SVHN, Fashion-MNIST and KMNIST both models are converged, at 1.75 and 0.13, 2.26 and 0.70, and 1.77 and 0.50; on CIFAR-100 neither is, at 6.21 for degree 1 and 3.23 for degree 2, so that dataset's Δ in Table 1 is an upper bound for

the same reason CIFAR-10's is. Every model in the second group is either degree 1 at one bit, or too narrow to fit (`A2d1R0w20` at width 20), or the fixed all-ones pair model, whose pair term is not learned. Three consequences. The gaps of Section 4.2 are upper bounds, since each is taken against a baseline that has not finished. The exchange of Section 4.3 is not, since all three of its comparisons lie inside the converged band, as do the three-bit pair, the degree-3 control and the SVHN pair, where the degree-1 model converges at 1.75 pp. The all-ones pair control is the mixed case: it holds width fixed and changes one variable, so its null is sound, but both of its models are outside the band, so it says that fixed all-ones pairs do not deliver the gain at a budget where neither model has finished, and it does not separate expressivity from conditioning. `A1d2R1w64` is the highest of the first group at 2.57 pp and should be read as marginal. The diagnostic is relative to this schedule and is not a proof of convergence.

Both headlines at a doubled budget. The upper bound above is testable rather than permanent. All four models of Section 4.2 were retrained at 240 epochs with the cosine schedule stretched to match, shared rate, three seeds. At matched parameters the baseline reaches 85.93 ± 0.10 and the pair unit 87.42 ± 0.13, an edge of $+1.50$ pp with disjoint seed bands; at matched logic, 84.94 ± 0.12 against 86.39 ± 0.08, an edge of $+1.45$, also disjoint. All four residuals are inside the band, at 1.43, 0.44, 1.82 and 0.58, the lower block of Table 9, and neither comparison is an upper bound any more: at 120 epochs the degree-1 models sit at 5.56 and 6.15 and the comparison is read against models that have not finished, and at 240 they do not. The budget is worth $+1.08$ and $+1.18$ pp to the two degree-1 models and $+0.46$ and $+0.43$ to the two degree-2 models: the narrowing follows the model and not the matching, and the two headlines of Section 4.2, $+2.11$ and $+2.20$, meet at about $+1.45$ once both models are trained out. The float reference was retrained for the same reason, a gap at one budget being unable to borrow a reference from another: it reaches 89.89 ± 0.22, gaining 0.33 where the baseline gains 1.08, which puts binarization's cost at 3.96 pp here against 4.72 at 120 epochs and the pair term returns 38 percent of it against 45. The tables stay at 120 epochs, the budget every model in this paper shares. This is the check that their ordering is not an artifact of stopping early, not a second set of results.

We keep the shared rate as the protocol because it fixes one variable across every model, and we report this sweep rather than re-anchoring the tables on it: a per-model sweep of every configuration in Table 2 was beyond the compute available. Every accuracy in Sections 4.2 and 4.3 is therefore a shared-rate number unless a rate is named, the two degree-1 configurations of the exchange 2×2 (`A1d1R0w122` and `A2d1R0w110`) included, that 2×2 being a second activation bit crossed with the pair term at near-equal logic (Section 4.3). The stored-bit counts do not move with the rate, since they follow width, which the budget fixes. The exchange is stated per accuracy point, whose denominator does move, and the two configurations should not move alike. `A2d1R0w110` is inside the band, and tuning it nonetheless lifted it 1.23 pp, widening the second bit's lead over the pair term to $+1.81$ pp and reversing the pair term's lead on top of that bit to -0.75 pp; being converged did not limit the lift. `A1d1R0w122` is outside it, which leaves the interaction of that 2×2 the quantity most exposed here. Its remaining configurations are reported in Table 10. One model was added after the sweep: the degree-1 baseline with the integer head at its own best rate, 4×10^{-3}, keeping the linear readout. It gains 0.87 pp from the head and 1.83 pp from the rate, and taking those as additive would put it at 87.54 and reverse the integer-head edge of Section 4.2. Measured, it reaches 86.56 ± 0.20, and the last row of Table 8 shows where the missing point went: the rate is worth $+1.83$ pp to the baseline with its float head and only $+1.00$ once the head is ternary, so the head has already bought most of what the rate was buying and the two cannot be added. The edge narrows to $+0.35$ pp against the pair unit at the same head and readout rather than reversing. The seed bands touch, so that is a narrowing and not a separation; this model has the largest residual in the study at 5.36, so $+0.35$ is itself an upper bound; and the fully integer corner, which replaces the readout as well, has not been run at a tuned rate.

The learnable activation range, and why the tables do not use it. A learnable activation range, one clipping scalar per block trained with the weights and five parameters in all, is a natural alternative to the fixed $[0, 1]$ clamp: post-BatchNorm values overflow that clamp, so with a fixed range some quantizer levels land where the data is not. It is set aside from the tables because it cannot be applied uniformly. At one bit there is no range to learn, so giving it to the upper rungs alone would move two variables at those steps. The tables therefore report the fixed clamp everywhere, and the comparison with the range is made here, where it is the more instructive of the two. With the range the three-bit

Table 10: The near-equal-logic comparison at each configuration's own best rate. Residual is what the configuration still gains over its last 60 epochs; the converged band is 2.57 or below. Only the base configuration is unconverged, and it does not enter the comparison between the repairs, which is the second row against the third.

configuration	unit	shared rate	best rate	at its best	residual
base	degree 1, one bit, `w122`	83.76	4×10^{-3}	85.34 ± 0.15	4.59
the second bit	degree 1, two bits, `w110`	86.54	4×10^{-3}	87.77 ± 0.16	1.96
the pair term	degree 2, one bit, `w40`	85.96	4×10^{-3}	85.97 ± 0.24	2.18
both	degree 2, two bits, `w36`	87.02	2×10^{-3}	87.13 ± 0.35	1.65

degree-1 model reaches 89.54 ± 0.23 and the degree-2 model 88.90 ± 0.23, an edge of -0.64 pp; without it they are 87.85 ± 0.19 and 87.93 ± 0.24, an edge of $+0.09$ pp. The range is worth 1.69 pp to degree 1 and 0.97 pp to degree 2, and that difference is the whole sign change. The same holds one rung down, which is what makes it a property of the range rather than of three bits: giving the two-bit pair the range as well moves them from 86.54 and 87.02 to 88.18 ± 0.08 and 87.90 ± 0.09, worth 1.64 pp to degree 1 against 0.88 pp to degree 2, and the edge from $+0.48$ pp to -0.28 pp. At both precisions the extra equipment is worth about three quarters of a point more to the additive unit, which is enough to turn a positive degree edge negative; the degree-1 model there is also still gaining, so -0.28 pp is an upper bound. Closing the remaining distance to the float reference with equipment of this kind is a separate question from which repair the unit should carry, and the one measured here is the second.

The exchange under tuning. The near-equal-logic 2×2 of Section 4.3 was measured at the shared rate, where three of its four configurations are degree-1 or one-bit models that the shared recipe does not finish. Giving every configuration its own best of 10^{-3}, 2×10^{-3} and 4×10^{-3}, three seeds each, moves the two sides very differently.

Table 10 gives the configurations. Over the tuned base the second bit is worth $+2.43$ pp and the pair term $+0.63$ pp, so the second bit leads by 1.80 pp with the two bands separated by 1.40, against a 0.58 pp lead at the shared recipe. Applying both is worth $+1.79$ pp, less than the bit alone, so once a second bit carries magnitude the pair term costs the width it is paid for; the interaction is -1.27 pp. The reason the shared recipe read the other way is in the last column but one: the two degree-1 configurations gain 1.58 and 1.23 from tuning and the two degree-2 configurations gain 0.01 and 0.11, so the shared recipe was not neutral between the repairs. Only the base configuration is still outside the converged band, and it does not enter the comparison between the repairs, which is the second row against the third; both of those are converged. What tuning does not change is the speed: at each model's own best rate the pair unit reaches 95 percent of its plateau at epoch 26 and the baseline at 47, against 28 and 49 at the shared rate (Section 4.2).

H The fan-in and rank grid, and the path to float

The grid holds sixteen configurations over fan-in and rank, trained on the shared recipe of Section 4.1 at three seeds each, with float weights and binary activations so that the two axes are read without the weight quantizer in the way; none of its numbers enters the tables of Section 4. Figure 6 draws both axes on the same accuracy scale, where the two are plainly unequal. Fan-in dominates: $+27.94$ pp from 9 to 216 inputs per unit at rank 1, against $+2.86$ pp for the whole rank ladder from one product to 48 at full fan-in, less than the $+7.57$ pp of the single fan-in step from 72 to 216. At full fan-in a single product already leads degree 1 by $+4.38$ pp, and rank shows geometrically diminishing returns from there, $+1.84$, $+0.86$ and $+0.16$ pp for $R = 1 \to 4 \to 16 \to 48$. Every rank trained at full fan-in sits below the bound of Corollary 3.5(b), which asks $R \geq 98$ already at $n = 216$ against a largest trained rank of 48. Degree 3 buys $+0.54$ pp over degree 2 at matched fan-in on this grid.

The shorter budget. Three results in this appendix and one in Appendix F predate the grid above and come from a 40-epoch recipe that flips whole batches rather than the shared one; they are named as such wherever they appear, and none enters the tables of Section 4 either. On that shorter budget

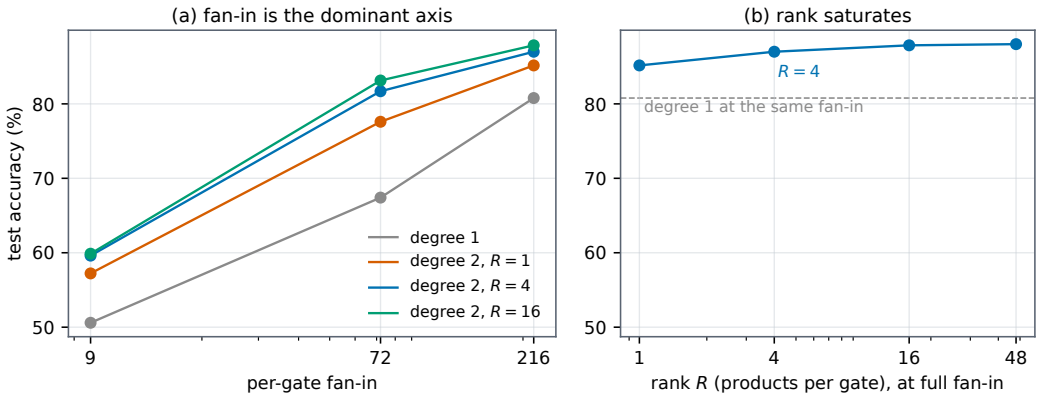

Figure 6: **Why full fan-in and why four products.** Both panels come from the grid of this appendix, with float weights and binary activations, on the shared recipe at three seeds per cell. **(a)** Accuracy against per-gate fan-in at four settings of the unit; the ordering by degree holds at each of the three fan-ins. **(b)** Accuracy against rank at full fan-in, with degree 1 at the same fan-in dashed. Fan-in moves accuracy far more than rank does, which is why the unit takes the full window and only four products. How the chosen unit trains at the shared recipe is Figure 2(a).

degree 3 loses 4.6 to 7.0 pp to degree 2 at matched AND-unit count in the pair form, over seven rank and fan-in pairings, one seed, sparse wiring. That is the width reading of the degree-3 result in Section 4.4. The optimizer reading is not excluded: at initialization that model's triple factors receive $7.2\times$ the gradient of its pair factors, and a per-branch correction lands 0.60 pp below the uncorrected model at the shared learning rate, 10^{-3}, at one seed.

Approach to the float reference. The reported models of Table 2 trace the path to float. At width 64 the degree-2 network gains from the second bit, 86.96 to 87.94, and nothing from the third, 87.93; the width-192 degree-1 network goes from 84.84 at one bit to 87.85 at three. Under the fixed clamp neither reaches the float reference. Of the 4.72 pp binarization costs, the pair term returns 2.11 pp at one bit and bits return the rest.

Degree. Degree 3 at matched parameters scores 0.81 pp below degree 2 on the shared recipe of Section 4.1, a direction. Matching parameters costs it width, 42 against 64, and fan-in with it, so the model separates degree from neither width nor fan-in; an optimizer cause is not excluded either, and both readings are given above.

Where degree stops mattering, in full. Three boundaries, all measured. *Precision*: the pair term's return falls from $+2.20$ pp at one bit to $+0.48$ pp on top of a second, is negative once that model is tuned, and is -0.64 pp at three bits. At three bits neither model reaches the float reference and the degree-1 model is still gaining, so the three-bit return is an upper bound; replacing the fixed clamp with a learnable quantizer range reverses its sign (Appendix G). *Fan-in*, the sharpest: it is worth $+27.94$ pp from 9 to 216 inputs per unit where the whole rank ladder spans less than one step of it (the grid above). What a product buys is not monotone in fan-in: the degree gap is $+9.02$ pp at 9 inputs per unit, $+14.29$ at 72 and $+6.22$ at 216, largest in the middle of the range measured and positive at both ends. The ordering by degree therefore survives the smallest fan-in as well as the largest, which is what the boundary claim needs; how the gap varies between them is not something these three points settle. *Tuning*: the matched-parameter gap falls to $+0.38$ pp once the baseline has its own rate. So degree matters where one is pinned to one bit, has real fan-in, and spends a fixed budget under a shared recipe. *Depth*: a degree-1 network at depth 7 and width 181, matched to both models on parameters within one percent, reaches 85.93 ± 0.16, which is 1.09 pp over the depth-5 baseline and 1.03 pp under the pair unit with the seed bands disjoint. Two extra blocks recover about half of what the pair term returns, so depth is a partial substitute and not a full one at this budget; that is one depth point and not a sweep.

I Protocol

Terms. A *model* here is one choice of activation bits, degree, rank and width, trained under the shared recipe with three seeds. The setting is one-bit activations, the binarized convolution of the title; the two- and three-bit models are the ablation along the precision axis (Sections 4.3 and 4.4), run under the same protocol so that the cost of leaving one bit can be compared with the cost of the pair term.

Training details of the unit. In a convolutional layer the unit reads the flattened patch x of Section 2. Parameters and training MACs both scale as branches times width squared, so at $R = 4$ width 64 matches degree 1 at width 192 in both ($9 \times 64^2 = 192^2$), to within 8 percent network-wide. Training uses AdamW at weight decay 5×10^{-4}, batch size 128 and a cosine schedule annealed over the epoch budget, with *crop and flip* meaning a random 32×32 crop from a reflection-padded 40×40 image and a random horizontal flip, applied per sample. At train time w, a_r, b_r are float, ternarized in the forward pass to $\{-1, 0, +1\}$, with the threshold at 0.7 times the filter's mean weight magnitude and a straight-through gradient. The output carries no per-filter scale; activations use the same estimator. The readout is a float linear layer by XNOR-Net convention, with an integer readout measured alongside (Section 4.2). From here on, wherever a unit is compiled, w, a_r, b_r and Q denote the ternarized tensors, and y denotes the integer the adder tree delivers to the comparison, $y = g(x) - \beta$ (Section 3.3, Figure 1).

The stack. At $w = 64$ the five blocks give per-unit fan-ins of 216, 576, 576, 1,152 and 1,152. They carry ternary weights, binary activations, BatchNorm folded at deployment, and a float linear readout except where stated. The stack is custom: its first layer reads binary planes and its readout admits an integer head, where XNOR-Net keeps both in float. Degree 1 is the ternary-weight popcount unit of the XNOR-Net and ternary-weight family. Degree 2 is the unit of Definition 3.1, with $R = 4$ products per unit unless stated. The replication datasets use the same network, front end, recipe and seeds as CIFAR-10; only the dataset and the readout's class count change, to 10 for SVHN and 100 for CIFAR-100, which leaves the matched-parameter pair matched there too (5.10M against 5.13M on CIFAR-100). The precision models are the codebase's two- and three-bit activation quantizers, with the same ternary weights and recipe. Training accuracy, where quoted, uses the first 10,000 training images without augmentation.

Hardware. Runs were distributed across NVIDIA A100 (40 GB), RTX A6000 and A40 GPUs, one model to a GPU, with no multi-GPU training anywhere in this paper. Which GPU a run drew was set by cluster availability and not by the model, so the three seeds behind a reported mean were not necessarily trained on the same device; we do not control for that, and it is one source of the seed spread reported with every number. Wall-clock cost of one model at the shared recipe ranges from about 50 minutes on CIFAR-10 to about 140 minutes on SVHN on an A6000, and roughly a third of that on an A100.

What a matched comparison holds. A matched comparison fixes a budget and lets width follow the degree; the degree-2 model is never the wider network. *Matched parameters* holds the parameter count and the training MACs to within 8 percent. *Matched logic* holds the weight-shared LUT-equivalent of Section 3.5 instead.

Cost-matching detail. The matched-parameter comparison sits at $1.028\times$ the baseline's weight-shared logic, and the per-tap charge is model-neutral: ternary zero fractions are 44.7 and 44.0 percent. `A1d1R0w122` sits within 0.01 of the width curve through the other one-bit degree-1 models. On the unrolled view `A1d2R4w40` carries 11.7 percent more logic than `A1d1R0w122`, worth about 0.2 pp, which leaves about $+2.0$ pp there.

What the degree-1 model is, and is not. It is this stack at degree 1, not a reimplementation, and its absolute accuracy should not be read against published ones. Rastegari et al. (2016) report 10.17 percent error on CIFAR-10, that is 89.83 percent, with binary weights and binary activations on the BinaryConnect architecture, leaving the first and last layers unbinarized. Our stack binarizes the input to the first layer through a frozen thermometer code, which costs 2.59 pp of float ceiling on its

own, 89.56 against 92.15 on raw pixels, so our float reference sits below their binary result. Every comparison in this paper is within one codebase, where the unit is the only variable.

Published accuracies in this family, and why they are not comparators. For orientation, published CIFAR-10 accuracies in this family, each on its own backbone and recipe: XNOR-Net reports 89.83 percent with binary weights and activations on the BinaryConnect architecture (Rastegari et al., 2016); FINN reports 80.1 percent for its streaming network at one bit (Umuroglu et al., 2017); FracBNN reports 89.1 percent on a ResNet-20 with float first and last layers and activations at an effective 1.4 bits (Zhang et al., 2021). These are not comparators for the numbers in Section 4: each differs from our setup in backbone, input encoding, first- and last-layer precision and training recipe at once, so a cross-paper difference of a point cannot be attributed to any one of those, let alone to degree. Every comparison in Section 4 changes one variable inside one setup, whose float reference is 89.56 percent (above), so that a difference can be attributed. Connecting the two would require a published degree-1 recipe run inside our setup.