Angular Networks: Low-Bit Learning from Randomized Similarity Estimators

Ali Ahmed[1,2,*] **Saeid Pourmand**[1] **Rehan Farooq**[2] **Muhammad Awais**[2] **Alireza Aghasi**[1,*]

[1]Oregon State University [2]Information Technology University

[*]Corresponding authors: {ahmeali, alireza.aghasi}@oregonstate.edu

Abstract

We propose a new framework for low-precision neural networks based on angular geometry and randomized low-bit estimators. Rather than approximating Euclidean linear operations under limited precision, we reinterpret neural computation in terms of directional similarity and construct quantized random-feature estimators of cosine interactions. Our approach introduces *Angular Layers*, which replace standard linear transformations with low-bit projections that estimate angular similarity in the forward pass, while gradients are computed with respect to the underlying continuous cosine geometry. This decoupling enables stable optimization without straight-through estimators or heuristic surrogate gradients, even under aggressive quantization of both weights and activations. On the theory side, we introduce *G-Gradient* as a gradient proxy for angular layers and show that, in a one-layer planted model, a sufficiently small G-gradient can guarantee exact recovery of a target binary solution. We further prove uniform approximation and convergence guarantees that provide a rigorous path from tractable continuous optimization to exact recovery in the original discrete model. Empirically, we instantiate these ideas in *Angular Nets*, including low-precision variants of ResNet, Vision Transformers, and BERT-style encoders, and obtain strong performance across vision and language tasks at substantially reduced precision. Overall, our results suggest that preserving angular structure provides a principled foundation for low-bit deep learning.

1 Introduction

Modern neural networks derive much of their expressive power from repeated high-dimensional linear transformations executed in floating-point precision. While this design has enabled remarkable progress across vision, language, and multimodal learning, it also incurs substantial costs in memory, bandwidth, and energy. As models continue to scale, these costs increasingly limit deployment, especially in edge and low-latency settings, motivating extensive work on quantized and low-precision neural networks. Despite this progress, aggressively reducing precision remains difficult. In particular, quantizing activations, especially inside attention mechanisms, often leads to unstable optimization and severe performance degradation [21, 3]. Small perturbations in inner products can produce disproportionately large downstream effects, disrupting learned representations and making training highly sensitive to approximation error. Existing approaches typically address this difficulty through heuristic gradient surrogates such as straight-through estimators (STE) [17, 4], calibration rules [8], or carefully tuned scaling strategies [12]. While often effective at moderate compression, these methods do not directly address the geometric fragility of low-precision computation; a more comprehensive review of this literature is deferred to Appendix A.

In this work, we argue that the difficulty is not purely numerical, but geometric. Standard neural network layers are naturally formulated in Euclidean space, where magnitude and direction are tightly

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

coupled and where similarity is mediated by Euclidean inner products. In contrast, high-dimensional representations often exhibit a more stable angular structure: norms concentrate, while relative directions remain discriminative. This viewpoint is consistent with the well-known spherical-shell concentration of generic high-dimensional random data [22], where norms concentrate and variation is primarily angular (see Figure 1). This suggests that, for low-precision learning, preserving angular similarity may be more important than preserving exact Euclidean inner products. Our central thesis is that low-bit learning should be built on angular geometry rather than Euclidean geometry.

A key mathematical observation is that angular similarity admits a randomized low-bit representation. Let $u, v \in \mathbb{S}^{p-1}$, and let $g \sim \mathcal{N}(0, I_p)$. A classical result is the Grothendieck identity [1, 14]

$$\mathbb{E}\left[\operatorname{sign}(g^\top u)\operatorname{sign}(g^\top v)\right] = \frac{2}{\pi}\arcsin\big(\langle u, v\rangle\big). \quad (1)$$

Consequently, if $\mathcal{G} \in \mathbb{R}^{N\times p}$ is a standard Gaussian matrix with i.i.d. entries, then $\frac{1}{N}\big\langle \texttt{sign}(\mathcal{G}u), \texttt{sign}(\mathcal{G}v)\big\rangle \approx \frac{2}{\pi}\arcsin\big(\langle u, v\rangle\big)$, for sufficiently large N. Thus, correlations of binary random embeddings provide a Monte Carlo approximation to an angular kernel. In practice, this approximation is still valid even with much simpler random embeddings such as Rademacher [1]. This resembles randomized kernel approximation methods such as Rahimi & Recht [16], but with a key geometric distinction: random features typically approximate Euclidean kernels such as the radial basis function kernel, whose similarity is governed by squared Euclidean distance, whereas the kernel here is intrinsically angular and depends only on directional alignment.

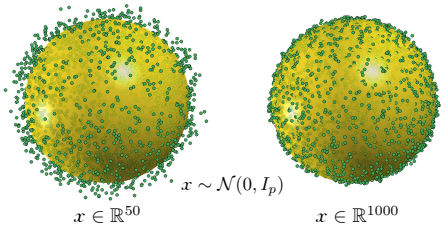


Figure 1: Spherical-shell concentration of random data. For low-dimensional visualization, each point preserves its original norm, while its 3D direction is obtained using principal component analysis.

Building on this perspective, we introduce *Angular Nets*, a family of low-precision neural architectures that depart from standard Euclidean networks by operating directly on directional structure. Their basic primitives are *Angular Layers*, which replace standard linear maps with randomized low-bit projections that estimate cosine interactions between normalized inputs and weights. Concretely, inputs and weights are normalized, projected onto random directions, quantized, and aggregated into a Monte Carlo estimator of angular similarity. In this sense, standard networks may be viewed as Euclidean nets, whereas Angular Nets are designed to compute and learn in angular geometry.

A central feature of Angular Nets is the decoupling of forward computation and backward optimization. The forward pass uses quantized random-feature estimators, producing a stochastic low-bit approximation of angular interactions, while the backward pass avoids differentiating through this estimator and instead computes gradients with respect to the underlying continuous cosine geometry. Thus, rather than approximating a discretized Euclidean computation directly, Angular Nets treat quantization as controlled estimator noise around a smooth angular objective, which is what enables stable training under aggressive quantization of both weights and activations.

This viewpoint also admits a concrete theoretical justification, by introducing the concept of G-gradients and Grothendieck gradient method (GGM) as an optimization flow for angular layers. The underlying binary recovery problem is combinatorial and intractable to optimize directly, yet in a one-layer planted model we show that vanishing, or even sufficiently small, G-gradient guarantees exact recovery of a target binary model under a projected null-space condition. We further prove uniform approximation of the G-gradient together with convergence guarantees implying that, under mild technical conditions, GGM reaches iterates whose G-gradient norm can be made arbitrarily small. Taken together, these results provide a rigorous bridge from tractable continuous optimization to exact recovery in the original discrete problem.

Empirically, we evaluate the framework from controlled low-bit experiments to full neural architectures. We instantiate Angular Nets by replacing linear layers with angular layers in convolutional and Transformer-based models, obtaining low-precision variants of ResNet, Vision Transformers, and BERT-style encoders, and show that preserving angular structure is sufficient to recover strong performance across modalities. Overall, our contributions are to introduce Angular Nets as a new family of low-precision architectures, develop corresponding layers and their decoupled training rule, connect the framework to Grothendieck-type identities and angular kernel estimation, establish recovery and convergence guarantees in the one-layer setting, and demonstrate strong empirical performance across vision and language tasks. Taken together, these results suggest a new design

principle for efficient deep learning: rather than approximating high-precision Euclidean operations directly, one can instead build neural networks in a geometry that is inherently compatible with low-bit representations.

Notations. Let $\mathbb{S}^{p-1}$ denote the unit sphere in $\mathbb{R}^p$. We use lowercase letters for vectors and uppercase letters for matrices. A random vector/matrix is called Gaussian (or Rademacher) if the elements are independently drawn from a standard normal distribution (or a Rademacher distribution taking values 1 and -1 with equal probability $1/2$). We use script font for random quantities (e.g., g, $\mathcal{G}$). We use teletype font for functions that operate on matrices/vectors in an element-wise fashion, e.g., $\texttt{sign}(x)$ is a vector with $\mathrm{sign}(x_i)$ as its i-th element.

2 Core Idea and Theory

We begin with a basic setting to isolate the main idea: inner products can be replaced by randomized low-bit angular interactions while preserving a smooth surrogate geometry for optimization. Since this replacement is local to each layer operation, the same construction extends naturally to multilayer architectures, including Transformer-style networks.

Linear Model Formulation. Suppose $x_i \in \mathbb{S}^{p-1}$ and $y_i \in \mathbb{R}$, for $i = 1, \dots n$. Let $\mathcal{Z} \in \mathbb{R}^{N\times p}$ be a given sign-embedding matrix of the input features. Our goal is to find a binary vector $w_b \in \{-1, 1\}^N$ such that $y_i \approx w_b^\top \texttt{sign}\left(\mathcal{Z}x_i\right)$, which amounts to solving the binary program

$$\underset{w_b\in\{-1,1\}^N}{\text{minimize}}\ \mathcal{L}_b(w_b) := \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{1}{N} w_b^\top \texttt{sign}\left(\mathcal{Z}x_i\right)\right)^2. \tag{2}$$

This is a combinatorially intractable problem. However, when $\mathcal{Z}$ is a Gaussian (or Rademacher) matrix, we can consider the parameterization $w_b = \texttt{sign}(\mathcal{Z}w)$ (assuming that the target binary solution is realizable, i.e., $w_b^\star = \texttt{sign}(\mathcal{Z}w^\star)$ for some $w^\star \in \mathbb{R}^p \setminus \{0\}$) and write

$$\mathcal{L}_b(w_b) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{1}{N}\texttt{sign}\left(w^\top\mathcal{Z}^\top\right)\texttt{sign}\left(\mathcal{Z}x_i\right)\right)^2 \approx \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{2}{\pi}\arcsin\left(\frac{w^\top x_i}{\|w\|_2}\right)\right)^2 = \mathcal{L}(w),$$

where in the approximation we used the Grothendieck identity. To perform the minimization using gradient-type methods, one can consider the discrete loss $\mathcal{L}_b(w_b)$ in the forward pass, and the continuous approximation $\mathcal{L}(w)$ in the back-propagation. The implication for this example is instead of using $\nabla\mathcal{L}(w)$ in a gradient descent framework, which delivers a stationary point of $\mathcal{L}(w)$, one may use an alternative proxy, so-called *G-gradient*, defined as

$$\begin{aligned}\tilde{\nabla}\mathcal{L}(w) &= -\frac{4}{n\pi}\sum_{i=1}^{n}\left(y_i - \frac{1}{N}\texttt{sign}\left(w^\top\mathcal{Z}^\top\right)\texttt{sign}\left(\mathcal{Z}x_i\right)\right)\frac{\partial}{\partial w}\arcsin\left(\frac{w^\top x_i}{\|w\|_2}\right)\\ &= -\frac{4}{n\pi}\sum_{i=1}^{n}\frac{\left(y_i - \frac{1}{N}\texttt{sign}\left(w^\top\mathcal{Z}^\top\right)\texttt{sign}\left(\mathcal{Z}x_i\right)\right)}{\sqrt{\|w\|_2^2 - (w^\top x_i)^2}}\left(x_i - \frac{w^\top x_i}{\|w\|_2^2}w\right).\end{aligned} \tag{3}$$

For Gaussian embeddings, it is immediate that $\mathbb{E}[\tilde{\nabla}\mathcal{L}(w)] = \nabla\mathcal{L}(w)$, so unlike STE-type methods, (3) provides an unbiased gradient proxy. Ideally, we would like the iteration $w_{k+1} = w_k - \eta_k\,\tilde{\nabla}\mathcal{L}(w_k)$, which we refer to as *Grothendieck gradient method (GGM)*, to converge to an iteration fixed point $\hat{w}$, whose binary embedding $\hat{w}_b = \texttt{sign}(\mathcal{Z}\hat{w})$ solves (2). We study this question in Section 2.1.

Extension to Multilayer Networks. The extension from the linear case to multilayer networks follows the same principle: the forward pass uses the discrete network loss, while the backward pass uses a Grothendieck-based gradient proxy. To formulate this, consider a depth-L network with hidden states $h_j = \phi_j(W_j{}^{(b)}h_{j-1})$, for $j = 1, \dots, L$, where $h_0(x) = x$ and $h_L(x) = f_b(x; \{W_j{}^{(b)}\}_{j=1}^L)$. The discrete training objective can be cast as

$$\mathcal{L}_b(\{W_j^{(b)}\}_{j=1}^L) = \frac{1}{n}\sum_{i=1}^{n}\ell\left(f_b(x_i;\{W_j^{(b)}\}_{j=1}^L), y_i\right) = \frac{1}{n}\sum_{i=1}^{n}\ell\left(f_b(x_i;\{\texttt{sign}(W_j\mathcal{Z}_j^\top)\}_{j=1}^L), y_i\right),$$

where analogous to the previous section we use the variable change $W_j{}^{(b)} = \texttt{sign}(W_j\mathcal{Z}_j^\top)$. Notice that the interaction of each binary row of $W_j{}^{(b)}$ with an incoming hidden representation is through

$\frac{1}{N_j}\texttt{sign}(w_j^\top \mathcal{Z}_j^\top)\texttt{sign}(\mathcal{Z}_j h_{j-1})$ which forms a binary layer operation when $\mathcal{Z}_j$ is Rademacher embedding. In the backward pass, such expressions are replaced by their $\arcsin$ surrogates $\frac{2}{\pi}\arcsin\big(w_\ell^\top h_{\ell-1}/(\|w_\ell\|_2\,\|h_{\ell-1}\|_2)\big)$. In this case the *G*-gradient takes the chain-rule formulation

$$\tilde{\nabla}_{W_\ell}\mathcal{L}(\{W_j\}_{j=1}^L) = \frac{1}{n}\sum_{i=1}^n \frac{\partial \ell(h_{L,i}, y_i)}{\partial h_{L,i}} \left(\prod_{j=1}^{L-\ell} \frac{\partial h_{L-j+1,i}}{\partial h_{L-j,i}}\right) \frac{\tilde{\partial} h_{\ell,i}}{\partial W_\ell},$$

where all forward quantities are computed using the discrete network and only the local Jacobian $\tilde{\partial} h_{\ell,i}/\partial W_\ell$ is computed using the $\arcsin$-proxy. The GGM scheme in this case follows the updates $W_\ell^{k+1} = W_\ell^k - \eta_k\,\tilde{\nabla}_{W_\ell}\mathcal{L}(\{W_j\}_{j=1}^L)$ for $\ell = 1, \ldots, L$.

2.1 *G*-Stationarity and GGM Convergence

Quality of *G*-Stationary Points. Running the GGM iteration $w_{k+1} = w_k - \eta_k\,\tilde{\nabla}\mathcal{L}(w_k)$, one seeks an iteration fixed point $\hat{w}$ satisfying $\tilde{\nabla}\mathcal{L}(\hat{w}) = 0$, which we call a *G*-stationary point for (2). A natural question is whether such points remain meaningful for the original discrete objective, given the mismatch between binary forward evaluation and continuous surrogate updates. Since $\mathbb{E}[\tilde{\nabla}\mathcal{L}(\hat{w})] = \nabla\mathcal{L}(\hat{w})$, for a fixed $\hat{w}$, for large embedding dimension N, this connection is partly explained by concentration: for Gaussian embeddings, the *G*-gradient concentrates around the gradient of the continuous angular loss. This consistency is formalized in the following result.

Proposition 2.1. *Let $x_1, \ldots, x_n \in \mathbb{S}^{p-1}$, let $w \in \mathbb{R}^p \setminus \{0\}$, and assume that w is not aligned with any x_i, i.e., $\|w\|_2^2 - (w^\top x_i)^2 > 0$, for $i = 1, \ldots, n$. Consider evaluating $\tilde{\nabla}\mathcal{L}(w)$ in (3) using a Gaussian embedding matrix $\mathcal{G} \in \mathbb{R}^{N\times p}$. For every $\varepsilon > 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$, if $N \geq \frac{32}{\pi^2\|w\|_2^2\varepsilon^2}\log\left(\frac{2n}{\delta}\right)$, then $\|\tilde{\nabla}\mathcal{L}(w) - \nabla\mathcal{L}(w)\|_2 \leq \varepsilon$.*

The preceding result justifies the *G*-gradient as a faithful proxy in the large-N regime, where randomized binary interactions concentrate around their angular expectations. Our main interest, however, is the genuinely low-bit regime, where N may be small and the discrete objective cannot simply be viewed as a smooth approximation. Notably, we show that, even in this setting, vanishing or near-vanishing *G*-gradient can certify recovery of a planted binary model $w_b^\star$. The argument proceeds in two steps: first, we establish deterministic exact recovery guarantees from a projected null-space condition; second, we show that this condition holds with high probability under natural assumptions on the geometry of the data and the resulting Gaussian sign process. To proceed with such analysis, let $\mathcal{G} \in \mathbb{R}^{N\times p}$ have i.i.d. Gaussian rows $g_1, \ldots, g_N \sim \mathcal{N}(0, I_p)$, and let $x_1, \ldots, x_n \in \mathbb{S}^{p-1}$ be fixed data vectors. Assume the observations are generated from an unknown planted binary vector $w_b^\star \in \{-1,1\}^N$ according to $y_i = N^{-1}(w_b^\star)^\top \operatorname{sign}(\mathcal{G}x_i)$ for $i = 1, \ldots, n$. For any $w \in \mathbb{R}^p \setminus \{0\}$, define $w_b := \operatorname{sign}(\mathcal{G}w) \in \{-1,1\}^N$ and $\delta := w_b - w_b^\star \in \{-2,0,2\}^N$. With the notation

$$\Pi_w^\perp := I_p - \frac{ww^\top}{\|w\|_2^2}, \quad \lambda_i(w) := \frac{1}{\sqrt{\|w\|_2^2 - (w^\top x_i)^2}}, \quad \mathcal{A}(w) := \sum_{i=1}^n \lambda_i(w)\,\Pi_w^\perp x_i\,\operatorname{sign}(\mathcal{G}x_i)^\top,$$

it is straightforward to verify from (3) that $\tilde{\nabla}\mathcal{L}(w) = \frac{4}{n\pi N}\mathcal{A}(w)(w_b - w_b^\star) = \frac{4}{n\pi N}\mathcal{A}(w)\delta$. Throughout this section we assume that $\Pi_w^\perp x_i \neq 0$ for $i = 1, \ldots, n$, which ensures that the quantities λ_i are well defined. Clearly, the null space of $\mathcal{A}(w)$ plays a central role because vanishing *G*-gradient is equivalent to the binary discrepancy vector $\delta = w_b - w_b^\star$ being annihilated by $\mathcal{A}(w)$. Therefore, if the restricted null space of $\mathcal{A}(w)$ contains no nonzero vector from $\{-2,0,2\}^N$, then $\tilde{\nabla}\mathcal{L}(w) = 0$ forces $\delta = 0$, and hence exact binary recovery follows. The following deterministic result formalizes this observation and applies independently of the distribution of the embedding matrix.

Proposition 2.2 (Recovery from vanishing *G*-gradient)**.** *Assume that $\Pi_w^\perp x_i \neq 0$ for all $x_i \in \mathbb{S}^{p-1}$, $i = 1, \ldots, n$. If $\ker(\mathcal{A}(w)) \cap \{-1,0,1\}^N = \{0\}$, and $\tilde{\nabla}\mathcal{L}(w) = 0$, then $w_b = w_b^\star$.*

Interestingly, the discreteness of the binary model induces a separation margin, so exact recovery can still occur when the *G*-gradient is sufficiently small rather than identically zero, as formalized next.

Proposition 2.3 (Recovery from small *G*-gradient)**.** *For $x_i \in \mathbb{S}^{p-1}$, assume that $\Pi_w^\perp x_i \neq 0$, and $\lambda_i(w) \geq \alpha_\lambda > 0$, for $i = 1, \ldots, n$ and $n \geq 4$. Further for $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{p\times n}$ assume $X^\top \Pi_w^\perp X \succeq \alpha_\Pi I_n \succ 0$. If $\ker(\mathcal{A}(w)) \cap \{-1,0,1\}^N = \{0\}$, and $\|\tilde{\nabla}\mathcal{L}(w)\|_2 < 16\alpha_\lambda\sqrt{\alpha_\Pi}/(n\pi N)$, then $w_b = w_b^\star$.*

The preceding propositions reduce recovery to a deterministic restricted null-space condition on $\mathcal{A}(w)$. Since $\mathcal{A}(w)$ is induced by the Gaussian embedding matrix $\mathcal{G}$, it only remains to show that this condition holds with high probability under suitable geometric assumptions on the data:

Theorem 2.1. *Assume that $\Pi_w^{\perp} x_i \neq 0$ for $x_i \in \mathbb{S}^{p-1}$, and let $X = [x_1, \dots, x_n]$. Assume that $0 \prec \alpha_L I_n \preceq X^\top X \preceq \alpha_U I_n$ and $X^\top \Pi_w^{\perp} X \succ 0$. Defining $\kappa_X = (2\alpha_U - \alpha_L)/(2 - \alpha_L)$, there exists a universal constant $C > 0$ such that for every $\beta \in (0,1)$, if $n \geq C\kappa_X \left(N/\log N + \log(N/\beta)\right)$, then $\mathbb{P}\left(\ker(\mathcal{A}(w)) \cap \{-1,0,1\}^N = \{0\}\right) \geq 1 - \beta$.*

Combined with Propositions 2.2–2.3, this shows that in the stated N-dimensional embedding regime, vanishing or sufficiently small G-gradient can certify exact recovery with high probability.

Convergence of GGM. The preceding section showed that G-stationarity is a meaningful certificate: in the large-N regime it is consistent with the continuous angular loss, while in the low-bit regime it can certify recovery of a planted binary model. We now turn to a separate question: whether the GGM iteration $w_{k+1} = w_k - \eta_k \tilde{\nabla}\mathcal{L}(w_k)$ can be guaranteed to reach such a point. Empirically, this iteration often stabilizes at a useful G-stationary point, but standard convergence arguments do not apply directly. The difficulty is twofold: the forward binary map is nonsmooth, and the vector field $\tilde{\nabla}\mathcal{L}(w)$ is not, in general, the gradient of an explicit potential function. To obtain a tractable convergence theory, we exploit the same Gaussian randomness used in the binary embedding through a Gaussian-smoothing construction, in the spirit of Nesterov's smoothing technique [13]. This leads to a smooth auxiliary potential $\mathcal{L}_\delta$ and a corresponding smoothed G-gradient g_δ, used primarily for analysis, and allows us to prove convergence to an approximate smoothed G-stationary point.

Proposition 2.4. *Define $\sigma_\delta(t) = \operatorname{erf}(t/(\delta\sqrt{2}))$, where $\operatorname{erf}(t) = 2/\sqrt{\pi}\int_0^t e^{-z^2}dz$ denotes the error function. If $u, v \in \mathbb{R}^p$ are fixed and g is a Gaussian vector, then $\mathbb{E}\left[\sigma_\delta(g^\top u)\,\operatorname{sign}(g^\top v)\right] = \frac{2}{\pi}\arcsin\left(\frac{u^\top v}{\sqrt{\|u\|_2^2+\delta^2}\,\|v\|_2}\right)$.*

With this smoothness, now consider

$$\mathcal{L}_\delta(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{2}{\pi}\arcsin\left(\frac{w^\top x_i}{\sqrt{\|w\|_2^2+\delta^2}}\right)\right)^2, \tag{4}$$

$$g_\delta(w) = -\frac{4}{n\pi}\sum_{i=1}^{n}\frac{\left(y_i - \frac{1}{N}\sigma_\delta\left(w^\top \mathcal{G}^\top\right)\operatorname{sign}\left(\mathcal{G}x_i\right)\right)}{\sqrt{(\|w\|_2^2+\delta^2) - (w^\top x_i)^2}}\left(x_i - \frac{w^\top x_i}{\|w\|_2^2+\delta^2}\,w\right), \tag{5}$$

where $\mathcal{G} \in \mathbb{R}^{N\times p}$ is a Gaussian matrix. We can guarantee convergence as stated in the next theorem.

Theorem 2.2. *Let $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$ be bounded below by $\mathcal{L}^\star \in \mathbb{R}$, and μ-smooth, that is $\forall w, w' \in \mathbb{R}^p$: $\mathcal{L}(w') \leq \mathcal{L}(w) + \nabla\mathcal{L}(w)^\top(w' - w) + \frac{\mu}{2}\|w' - w\|_2^2$. Assume moreover that for all $w \in \mathbb{R}^p$, $g : \mathbb{R}^p \to \mathbb{R}^p$ satisfies $\|\nabla\mathcal{L}(w) - g(w)\|_2 \leq \varepsilon$. Consider the iteration $w_{k+1} = w_k - \eta\, g(w_k)$, with stepsize $\eta \in (0, 1/\mu)$. Then $\min_{0\leq k\leq T-1}\|g(w_k)\|_2^2 \leq \frac{2\,(\mathcal{L}(w_0)-\mathcal{L}^\star)}{T(\eta-\mu\eta^2)} + \frac{\varepsilon^2}{(1-\mu\eta)^2}$ for every integer $T \geq 1$. In particular, setting $\eta = \frac{1}{2\mu}$ yields $\min_{0\leq k\leq T-1}\|g(w_k)\|_2^2 \leq \frac{8\mu\,(\mathcal{L}(w_0)-\mathcal{L}^\star)}{T} + 4\varepsilon^2$.*

In order to provide convergence guarantees for GGM using Theorem 2.2 we need to establish the smoothness and gradient consistency for $\mathcal{L}_\delta(w)$ as stated in (4). The following result shows that $\mathcal{L}_\delta(w)$ is globally $\mathcal{O}(\delta^{-2})$-smooth.

Proposition 2.5. *Consider $\mathcal{L}_\delta(w)$ as stated in (4), with $x_i \in \mathbb{S}^{p-1}$. Then $\mathcal{L}_\delta(w)$ is $\mu_{\mathcal{L}}$-smooth, where $\mu_{\mathcal{L}} = \frac{8}{\pi^2\delta^2} + \frac{16}{n\pi\delta^2}\sum_{i=1}^{n}(|y_i|+1)$.*

In the next result we will also show that thanks to the bounded nature of g_δ and $\nabla\mathcal{L}_\delta$ we can uniformly bound $\|\nabla\mathcal{L}_\delta(w) - g_\delta(w)\|_2$, which completes the ingredient required to apply Theorem 2.2.

Theorem 2.3. *Consider $\mathcal{L}_\delta(w)$ and $g_\delta(w)$ as stated in (4) and (5), where $x_i \in \mathbb{S}^{p-1}$ and $\mathcal{G} \in \mathbb{R}^{N\times p}$ is a Gaussian matrix. Then, there exists a universal constant $C > 0$ such that for every $\beta \in (0,1)$: $\sup_{w\in\mathbb{R}^p}\|\nabla\mathcal{L}_\delta(w) - g_\delta(w)\|_2 \leq C\delta^{-1}(\sqrt{p/N} + p/N + \sqrt{\log(2n/\beta)/N})$, with probability at least $1-\beta$.*

An immediate corollary follows by combining Theorem 2.3 and Proposition 2.5 with the convergence result in Theorem 2.2. In particular, for a fixed smoothing parameter $\delta > 0$, choosing $N = \Omega(\frac{p+\log n}{\delta^2\varepsilon})$ identifies an iterate satisfying $\min_{0\leq k\leq T-1}\|g_\delta(w_k)\|_2^2 \leq \varepsilon$ within $T = \Omega(\frac{1}{\delta^2\varepsilon})$ iterations.

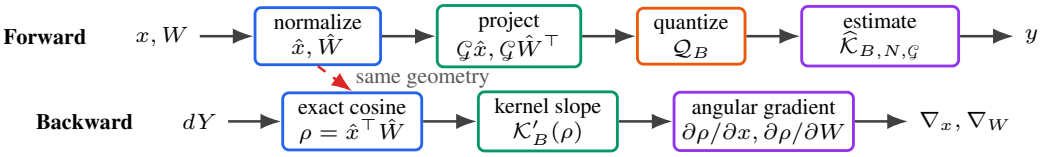


Figure 2: Angular Layer: low-bit random features estimate $\mathcal{K}_B(\rho)$ in forward pass, while exact cosine geometry provides the backward signal, avoiding differentiation through the quantized estimator.

3 Practical Anatomy of GGM

Implementation of an Angular Layer. We now summarize the forward and backward implementation of an angular layer. For $x, w \in \mathbb{R}^p$, define the normalized vectors $\hat{x} = x/\|x\|_2$, $\hat{w} = w/\|w\|_2$, and their angular correlation $\rho = \hat{x}^\top \hat{w}$. Unlike standard low-precision layers, our approach does not quantize the Euclidean inner product $x^\top w$ directly. Instead, it maps normalized directions through random projections $\mathcal{G} \in \mathbb{R}^{N\times p}$, where $N = \nu p$ (we refer to ν as the projection expansion ratio). It applies a symmetric B-bit quantizer $\mathcal{Q}_B$, and computes

$$\widehat{\mathcal{K}}_{B,N,\mathcal{G}}(\rho) = N^{-1} \sum_{i=1}^{N} \mathcal{Q}_B(g_i^\top \hat{x}) \mathcal{Q}_B(g_i^\top \hat{w}), \tag{6}$$

which estimates the angular kernel $\mathcal{K}_B(\rho) = \mathbb{E}[\mathcal{Q}_B(u)\mathcal{Q}_B(v)]$, where (u, v) are standard Gaussian variables with correlation ρ. For $B = 1$, $\mathcal{Q}_1(z) = \text{sign}(z)$, and Grothendieck's identity gives $\mathcal{K}_1(\rho) = \frac{2}{\pi}\arcsin(\rho)$. Thus, the low-bit computation estimates an angular kernel rather than the Euclidean correlation itself.

The backward pass follows the corresponding smooth angular geometry instead of differentiating through the quantized estimator: $\nabla_w \mathcal{K}_B(\rho(w, x)) = \mathcal{K}'_B(\rho)(\hat{x} - \rho\hat{w})/\|w\|$. Here, $\hat{x} - \rho\hat{w}$ is the component of the input direction orthogonal to the current weight direction, so the update rotates w toward better angular alignment rather than applying a coordinatewise correction to a hard threshold. Figure 2 illustrates this forward/backward decoupling: the forward pass uses quantized random features, while the backward pass uses the exact cosine geometry and smooth kernel slope.

This fundamentally differs from STE-based binarization. STE methods begin with an element-wise hard quantizer $\mathcal{Q}_B$ and prescribe a surrogate derivative, e.g., $\mathcal{Q}'_B(z) \approx 1$, yielding coordinatewise updates tied to a scalar threshold approximation. GGM instead differentiates a continuous angular objective, coupling all coordinates through the normalized correlation ρ and producing a smooth rotational vector field. Figure 3 visualizes this contrast, where GGM adjusts angular alignment, while a representative STE update remains more axis-separable.

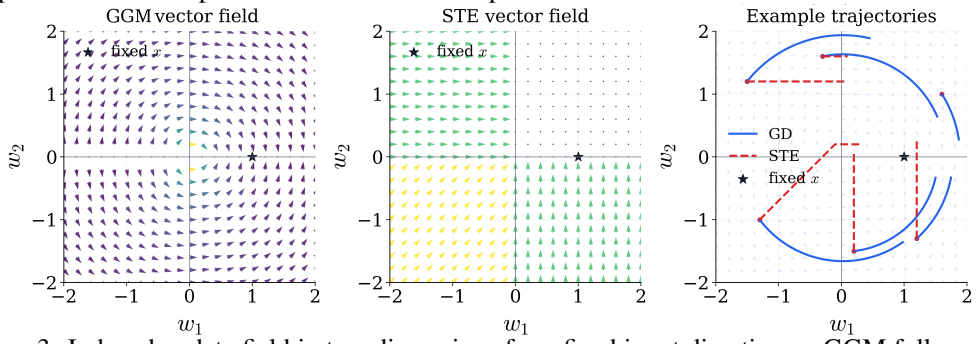


Figure 3: Induced update field in two dimensions for a fixed input direction x. GGM follows the gradient of the angular kernel $\mathcal{K}_B(\rho(w, x))$, producing rotational updates that adjust cosine similarity. A representative STE produces a more coordinatewise, axis-separable update pattern.

A Natural Generalization Beyond Binary Systems. A key advantage of the proposed framework is its natural extension beyond binary systems through a generalization of Grothendieck's identity beyond sign functions (see Section J of Appendix). The multibit case preserves the same angular interpretation with a bit-dependent kernel $\mathcal{K}_B$. As shown in Figure 4(a), for unit-normalized vectors, the vector-gradient magnitude is proportional to $\mathcal{K}'_B(\rho)\sqrt{1-\rho^2}$, which concentrates learning on intermediate correlations and decays smoothly as $|\rho| \to 1$.

Rademacher vs. Gaussian Embeddings and Seed Compression. Although the theory is stated for Gaussian embeddings, in practice we observe nearly identical performance with Rademacher

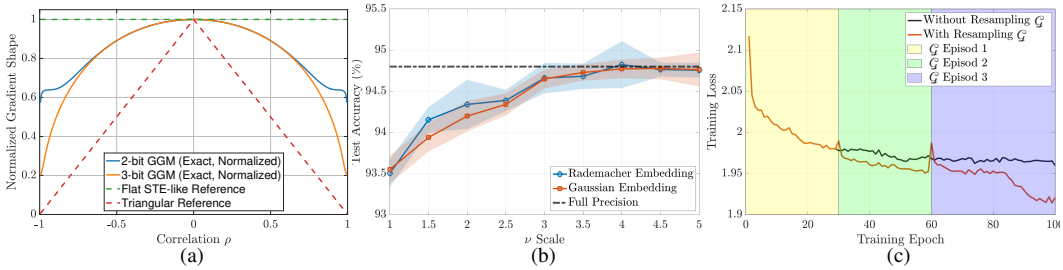


Figure 4: (a) 2-bit and 3-bit GGM gradients are smooth functions of angular correlation ρ, unlike coordinatewise STE-style references. (b) Accuracy of a binary GGM ResNet-18 model trained on CIFAR-10 using Gaussian and Rademacher embeddings across projection expansion ratios $\nu = N/p$. The two embeddings yield similar performance. (c) Training loss of a binary convolutional neural network (CNN) with and without resampling the random embedding matrices. Although resampling $\mathcal{G}$ causes transient loss spikes, it can help the optimizer escape unfavorable basins and converge to a lower loss than a fixed-$\mathcal{G}$ run.

embeddings, which are substantially simpler to generate and store (see Figure 4(b)). This is consistent with the result of Aghasi *et al.* [1], who showed that for spread-out unit vectors u, v, i.e., $\|u\|_\infty, \|v\|_\infty = \mathcal{O}(p^{-1/2})$, and a Rademacher vector $\imath$, the Grothendieck identity continues to hold up to an $\mathcal{O}(p^{-1/2})$ error. Appendix K provides a more detailed summary of alternative forms of Grothendieck's identity for embedding matrices beyond Gaussian. Moreover, the embedding matrix in an angular layer need not be stored explicitly: since it is a generic random object, it can be regenerated from a random seed, making the effective storage overhead negligible.

More Exploratory Landscape Search. The random projection matrix $\mathcal{G}$ introduces a second source of stochasticity beyond minibatch sampling. Fixing $\mathcal{G}$ optimizes one frozen random-feature approximation of the angular kernel, while resampling $\mathcal{G}$ exposes the model to different binary sketches of the same target geometry. Similar in spirit to noise-driven exploration in Langevin dynamics [23, 15], this stochasticity can perturb the trajectory away from a current basin and encourage a more exploratory search of the loss landscape. Although resampling may create transient loss spikes, these perturbations can help the optimizer escape unfavorable basins and converge to better solutions (see Figure 4(c)). Additional benefits of randomness in angular layers, including improved model robustness, are discussed in the experiments.

4 Experiments Overview and Discussion

We evaluate GGM across a broad range of modalities, architectures, and layer types to test whether its effectiveness reflects a general principle for low-bit learning rather than a benchmark-specific design. Our experiments span convolutional image models on CIFAR-10, TinyImageNet, and ImageNet-1K, vision Transformers, Transformer encoders for natural language processing (NLP), and Transformer-based time-series models (e.g., Tables 1 and 2). Across these settings, GGM is applied to the dominant linear operators of the network, including convolutional layers, fully connected layers, and attention/MLP (multilayer perceptron) projections. This breadth is central to our evaluation philosophy: rather than specializing GGM to a single application regime, we ask whether one geometric low-bit mechanism can remain competitive across many of them. Additional experiments and implementation details are presented in Section L of Appendix. Also, our implemented models are available at: `https://github.com/CACTuS-AI/AngularNet`.

Controlled CIFAR-10 ViT Binarization Study We begin with a controlled CIFAR-10 ViT study because Transformer-style vision models are a difficult regime for fully binary layers: errors in attention and MLP projections propagate through token mixing, normalization, and residual pathways. Rather than reproducing full method-specific pipelines, which may include distillation [11] or multi-stage optimization [20], we isolate layer-level binarization mechanisms under a fixed ViT backbone and identical training recipe.

We denote low-bit configurations by $(\nu_w/\nu_a) \times \nu$, where ν_w and ν_a are weight and activation bit-widths, and ν is the projection expansion ratio ($\nu = N/p$). For example, $(1/1) \times 4.0$ denotes binary weights/activations with $N = 4p$. Direct STE layers have no analogous projection dimension, and increasing their binary capacity requires widening the Euclidean network before binarization. We

therefore report the controlled STE variants at $(1/1) \times 1.0$, and study STE width scaling separately in Figure 5.

All controlled STE baselines apply a direct quantized linear map $y = \gamma\, \mathcal{Q}_{B_x}(x)\mathcal{Q}_{B_w}(W)^\top$, where $\mathcal{Q}_{B_x}, \mathcal{Q}_{B_w}$ denote the B_x and B_w bit activation and weight quantizers, respectively, including any method-specific scale, offset, or periodic parameterization, and γ is an optional learnable channel-wise output gain. In the controlled study we use clipped STE [2], $\partial\mathcal{Q}(z)/\partial z \approx 1_{|z|\le s}$. Depending on the method, activation scales are computed per sample/token while weight scales are scalar or per output channel. We compare against six binarization baselines. STE-AMAX uses $\mathcal{Q}_1(z) = s\,\mathrm{sign}(z)$ with $s = \max|z|$ and no gain, while STE-AMAX+G adds gain to the same quantizer. The XNOR+G baseline instead uses $s = \mathrm{mean}(|z|)$, following XNOR-Net [17], while LSCALE+G introduces a learnable scaling factor through $s = \lambda\,\mathrm{mean}(|z|)$, following the idea of learned-scale quantization [5]. We also include ADABIN+G, which uses $\mathcal{Q}_1(z) = \beta + s\,\mathrm{sign}(z-\beta)$ [19], and BIPER-LF+G, which adopts $\mathcal{Q}_1(z) = s\,\mathrm{sign}(\sin(\omega z))$, inspired by BiPer [20]. For BIPER-LF+G, we fix $\omega = 2$, as this performed best in our experiments. Making ω learnable or using other fixed values led to instability after reaching moderate accuracy, likely due to the non-monotone periodic surrogate. We therefore report the best accuracy achieved before divergence.

GGM uses the angular RF estimator of Eq. (6). In the controlled 1-bit runs, we use one learnable scale per output channel, γ_o as $y_o = \gamma_o \frac{1}{N}\sum_{i=1}^{N} \mathrm{sign}(g_i^\top \hat{x})\mathrm{sign}(g_i^\top \hat{w}_o)$. For large N, this behaves like an effective scaled 1-bit kernel, $\mathbb{E}_{\mathcal{G}}[y_o] \approx \gamma_o \mathcal{K}_1(\rho_o)$, where $\mathcal{K}_1(\rho) = \frac{2}{\pi}\arcsin(\rho)$. For the 1.58-bit row, GGM uses the same estimator with symmetric ternary levels $\{-1, 0, +1\}$.

Table 1: Controlled CIFAR-10 ViT binarization study. All STE baselines use the same ViT backbone, optimizer, and training budget as GGM, and all use clipped STE. GGM uses projected-coordinate quantization with learnable output gain; in the binary case, s denotes XNOR-style learned scaling on the projected coordinates. The 1.58-bit setting denotes ternary quantization, since $\log_2 3 \approx 1.58$. For each controlled baseline, we report the best preprocessing setting found.

Method	**Quantizer / Scale**	**Config.**	**Top-1**
ViT (FP)	FP (Full Precision)	32/32	92.49
STE-AMAX	$s\,\mathrm{sgn}(z),\ s = \max\|z\|$	$(1/1) \times 1.0$	57.52
STE-AMAX+G	$s\,\mathrm{sgn}(z),\ s = \max\|z\|$	$(1/1) \times 1.0$	58.64
XNOR+G	$s\,\mathrm{sgn}(z),\ s = \mathrm{mean}\|z\|$	$(1/1) \times 1.0$	78.96
LSCALE+G	$s\,\mathrm{sgn}(z),\ s = \lambda\,\mathrm{mean}\|z\|$	$(1/1) \times 1.0$	78.96
ADABIN+G	$\beta + s\,\mathrm{sgn}(z-\beta)$	$(1/1) \times 1.0$	79.39
BIPER-LF+G	$s\,\mathrm{sgn}(\sin(\omega z))$	$(1/1) \times 1.0$	77.12
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1) \times 1.0$	**86.51**
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1) \times 2.0$	**89.46**
GGM (Ours)	RF kernel, $\mathcal{Q}_{\log_2 3}(z) \in \{-s, 0, +s\}$	$(\log_2 3/\log_2 3) \times 1.0$	**90.59**
GGM (Ours)	RF kernel, $\mathcal{Q}_{\log_2 3}(z) \in \{-s, 0, +s\}$	$(\log_2 3/\log_2 3) \times 2.0$	**91.92**

In our experiments, GGM consistently outperforms all controlled STE baselines under identical settings by a substantial margin. Notably, it operates in a distinct performance regime, achieving near full precision accuracy even in the ternary setting with $\nu = 1$. Increasing ν further closes much of the gap to full precision, suggesting that the gain comes from GGM's angular random-feature structure rather than from scaling, output gain, or a stronger STE surrogate alone.

Table 2: Cross-domain results.

Domain	**Task / Method**	**Config.**	**Acc.**
NLP	SST-2 / FP	32/32	93.2
	SST-2 / BiBERT	$(1/1) \times 1.0$	88.7
	SST-2 / GGM	$(1/1) \times 1.0$	**90.56**
	QNLI / FP	32/32	92.1
	QNLI / BiBERT	$(1/1) \times 1.0$	72.6
	QNLI / GGM	$(1/1) \times 1.0$	**86.24**
Time	PTB-XL / FP	32/32	72.87
	PTB-XL / GGM	$(1/1) \times 1.0$	**73.08**
	ADFTD / FP	32/32	53.27
	ADFTD / GGM	$(1/1) \times 1.0$	**53.39**

We evaluate GGM on TinyImageNet and ImageNet-1K using 12-block, six-head ViTs with MLP ratio 4, respectively at resolution/patch size/width $64/4/192$ and $224/16/384$. Our scaffold uses normalized queries/keys, RMSNorm, OddGate, and mean pooling. We apply GGM to all 72 attention and MLP projections, enabling end-to-end evaluation of quantized transformers rather than selectively quantized layers. Experiments cover multiple quantization configurations, including $\nu = N/p = 1$, and compare against conventional full-precision ViT references using ordinary linear maps. Models are optimized using AdamW with training schedules adapted to the dataset and quantization set-

ting. This setup allows us to assess GGM across different image resolutions, model widths, and quantization regimes.

Table 3: TinyImageNet: selected EMA top-1 (%). FP denotes the conventional linear body.

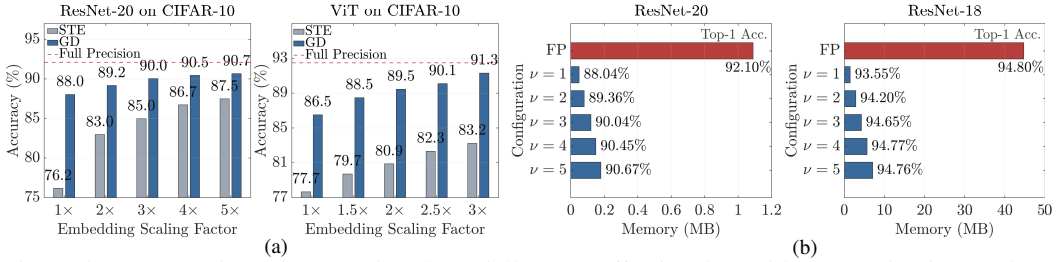
Network/Method	Config.	Top-1
ViT (FP)	$32/32$	55.10
LSQ-style	$(1/1) \times 1$	40.81
GGM (Ours)	$(1/1) \times 1$	**51.70**

Table 4: ImageNet-1K: selected EMA top-1 (%). Precision refers to the attention/MLP projections.

Network/Method	Config.	Top-1
ViT (FP)	$32/32$	79.342
LSQ-style	$(1/1) \times 1$	46.470
LSQ-style	$(2/2) \times 1$	73.594
GGM (Ours)	$(1/1) \times 1$	**66.834**
GGM (Ours)	$(\log_2 3/\log_2 3) \times 1$	**73.464**
GGM (Ours)	$(2/2) \times 1$	**76.128**
GGM (Ours)	$(\log_2 5/\log_2 5) \times 1$	**77.168**

GGM outperforms LSQ-style by 10.89 points on TinyImageNet and by 20.364/2.534 points on ImageNet-1K at $(1/1) \times 1$/$(2/2) \times 1$, respectively. Comparisons follow a consistent experimental protocol and are based on single runs without baseline-specific hyperparameter searches. At five levels, GGM approaches the performance of its corresponding full-precision formulation. For external context, our $(2/2) \times 1$ result exceeds Q-ViT's 72.1% on DeiT-S [10]. BHViT-Small reports a higher binary accuracy of 68.4%, using a convolution–Transformer hybrid designed for binarization [6]. In contrast, GGM improves low-bit projections within a ViT backbone without introducing convolutional token-mixing stages. Its layer-level construction could complement such hybrid architectures, although performance on those backbones remains untested. Differences in architecture, training protocols, and full-precision coverage prevent direct cross-paper rankings. Related comparisons with SI-BiViT and BiViT [24, 7] are also considered. CIFAR-10 ViT experiments are reported separately in Table 1, providing a controlled setting for evaluating layer-level binarization under a fixed architecture. Further experimental results and discussion are provided in Section L of the Appendix.

Figure 5 tests whether GGM improves merely by using more binary coordinates. The left panel compares GGM projection expansion against width-scaled STE baselines. Increasing ν in GGM yields substantially larger gains than widening STE models, indicating that the improvement is not explained by binary capacity alone. Table 5 summarizes convolutional results on CIFAR-10,



(a) (b)

Figure 5: (a) Increasing ν in GGM is substantially more effective than width expansion in STE baselines. (b) Representative GGM configurations retain a large memory advantage over full precision.

TinyImageNet, and ImageNet-1K. GGM improves consistently with ν, and larger-ν configurations are competitive with or stronger than specialized binary CNN baselines. Additional details are provided in Appendix L.3.

Table 2 shows that the same mechanism transfers beyond vision. On the NLP datasets SST-2 and QNLI, GGM improves over binarized Transformer baselines, with the largest gain on QNLI where the prior binary model degrades substantially. On medical time-series classification, GGM remains

competitive with the full precision Medformer baseline despite using 1-bit weights and activations in the replaced linear operators. Additional details are provided in Appendix L.6.

Table 5: CNN vision results. The FP / Top Priors column reports full precision and the two strongest single-stage 1-bit baselines. Two-stage methods such as BiPer are excluded from the main comparison and reported separately in Appendix Tables 7 and 8. For GGM, increasing ν improves accuracy while preserving low-bit computation.

Dataset	Architecture	FP / Top Priors	Config.	Acc.
CIFAR-10	ResNet-18	FP 94.8 1/1: AdaBin 93.10, ReCU 92.8	$(1/1) \times 1.0$ $(1/1) \times 2.5$	**93.58** **94.51**
	ResNet-20	FP 92.1 1/1: AdaBin 88.20, RBNN 87.80	$(1/1) \times 1.0$ $(1/1) \times 2.5$	**88.51** **89.74**
	VGG-16	FP 94.04 1/1: AdaBin 90.26, RBNN 88.13	$(1/1) \times 1.0$ $(1/1) \times 2.5$	**91.47** **93.12**
TinyImageNet	ResNet-18	FP 65.48 1/1: IR-Net 56.12, RBNN 55.79	$(1/1) \times 1.0$ $(1/1) \times 2.0$	**59.84** **63.48**
	ResNet-34	FP 67.85 1/1: IR-Net 60.20, RBNN 59.57	$(1/1) \times 1.0$ $(1/1) \times 2.0$	**64.46** **66.54**
ImageNet-1K	ResNet-18	FP 69.6 1/1: AdaBin 63.10, ReCU 61.00	$(1/1) \times 1.0$ $(1/1) \times 2.0$	**63.53** **66.39**
	ResNet-34	FP 73.3 1/1: AdaBin 66.40, ReCU 65.10	$(1/1) \times 1.0$ $(1/1) \times 2.0$	**68.82** **70.92**

In closing, we believe that Angular Nets and randomized embeddings open a broader direction for both theory and applications in low-bit learning. Our analysis of G-gradients for linear layers is only a first proof of concept, and extending this theory to attention, residual architectures, and deep compositional models raises new questions at the interface of optimization, random matrix theory, and high-dimensional probability. On the application side, the same angular low-bit mechanism suggests a promising path toward more efficient foundation models, including large language and multimodal systems. More broadly, this work points to random geometric computation as a principled foundation for future compressed neural architectures. We provide additional discussion of limitations, open research avenues, and broader impacts in Appendix M.

Acknowledgment

This work was partially supported by DARPA-DSO under Grant HR0011-26-3-E048. Alireza Aghasi acknowledges computational resources provided through allocation CIS261209 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation. He also acknowledges support from the Google Cloud Research Credits program.

References

[1] Alireza Aghasi, Nicholas F. Marshall, Saeid Pourmand, and Wyatt D. Whiting. G-net: A provably easy construction of high-accuracy random binary neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[3] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36:75067–75096, 2023.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[5] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020.

[6] Tian Gao, Yu Zhang, Zhiyuan Zhang, Huajun Liu, Kaijie Yin, Chengzhong Xu, and Hui Kong. Bhvit: Binarized hybrid vision transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3563–3572, 2025.

[7] Yefei He, Zhenyu Lou, Luoming Zhang, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Bivit: Extremely compressed binary vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5651–5663, 2023.

[8] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International conference on machine learning*, pages 4466–4475. PMLR, 2021.

[9] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang, and Qi Tian. Super-bit locality-sensitive hashing. *Advances in neural information processing systems*, 25, 2012.

[10] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: accurate and fully quantized low-bit vision transformer. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 34451–34463, 2022.

[11] Zechun Liu, Barlas Oguz, Aasish Pappu, Lin Xiao, Scott Yih, Meng Li, Raghuraman Krishnamoorthi, and Yashar Mehdad. BiT: Robustly binarized multi-distilled transformer. In *Advances in Neural Information Processing Systems*, volume 35, pages 14303–14316, 2022.

[12] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1325–1334, 2019.

[13] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[14] Saeid Pourmand, Wyatt D Whiting, Alireza Aghasi, and Nicholas F Marshall. Laplace-hdc: Understanding the geometry of binary hyperdimensional computing. *Journal of Artificial Intelligence Research*, 82:1293–1323, 2025.

[15] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703. PMLR, 2017.

[16] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[17] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[18] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[19] Zhijun Tu, Xinghao Chen, Pengju Ren, and Yunhe Wang. AdaBin: Improving binary neural networks with adaptive binary sets. In *European Conference on Computer Vision*, pages 379–395. Springer, 2022.

[20] Edwin Vargas, Claudia Correa, Carlos Hinojosa, and Henry Arguello. BiPer: Binary neural networks using a periodic function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6840–6849, 2024.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[22] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[23] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[24] Peng Yin, Xiaosu Zhu, Jingkuan Song, Lianli Gao, and Heng Tao Shen. Si-bivit: Binarizing vision transformers with spatial interaction. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 8169–8178, 2024.

Appendix

Contents

A A Review of Low-Bit Learning Literature **14**

B Proof of Proposition 2.1 **16**

C Proof of Proposition 2.2 **17**

D Proof of Proposition 2.3 **17**

E Proof of Theorem 2.1 **18**

E.1 Auxiliary Lemmas 18

E.2 Main Proof of Theorem 2.1 22

F Proof of Proposition 2.4 **22**

G Proof of Theorem 2.2 **23**

H Proof of Proposition 2.5 **23**

I Proof of Theorem 2.3 **26**

I.1 Auxiliary Lemmas 26

I.2 Main Proof of Theorem 2.3 28

J Natural Generalization Beyond Binary Systems **31**

K Grothendieck's Identity Beyond Gaussian Embeddings **32**

K.1 Gaussian Grothendieck's Identity 32

K.2 Rotational Invariance and Orthogonal Embeddings 32

K.3 Rademacher Embeddings 33

K.4 Full-Width Randomized Walsh–Hadamard Embeddings 34

L Experimental Details and Additional Results **34**

L.1 Layer Replacement Policy 34

L.2 Controlled CIFAR-10 ViT Study 35

L.3 Image Classification 35

L.3.1 CIFAR-10 Convolutional Models 36

L.3.2 TinyImageNet Convolutional Models 37
L.3.3 ImageNet-1K Convolutional Models 38
L.3.4 CIFAR-10 Vision Transformers 38
L.4 TinyImageNet vision Transformer Experiments 39
L.5 ImageNet-1K Vision Transformer Experiments 40
L.6 Transformer Text Classification 42
L.7 Medical Time-Series Classification 43
L.8 Additional Analyses 43
L.8.1 Projection Distribution: Gaussian vs. Rademacher 43
L.8.2 Resampling the Projection Matrix 44
L.8.3 Projection Expansion vs. STE Width Scaling 44
L.8.4 Memory–Accuracy Trade-off 45
L.8.5 Robustness to Binary Weight Perturbations 46
L.9 Gradient–Correlation Profiles for GGM Kernels 46
L.10 End-to-End Efficiency of Packed Angular Vision Transformers 47
L.10.1 CIFAR-10 47
L.10.2 ImageNet-1K 51

M Limitations and Broader Impacts **54**

A A Review of Low-Bit Learning Literature

Quantization is one of the most widely studied approaches to reducing the memory, bandwidth, and computational cost of deep neural networks. Early work demonstrated that full precision networks contain substantial numerical redundancy, motivating the replacement of floating-point weights and activations with lower-bit representations [4, 17], [33, 68, 36]. Subsequent research refined this idea along several axes: ternary and low-bit weight quantization [40, 69, 67], quantization-aware training with learned clipping thresholds, step sizes, or quantization intervals [2, 5], [66, 37, 32], hardware-aware and mixed-precision schemes [59, 30, 29, 63], and post-training methods that adapt rounding or reconstruct local layer outputs without full retraining [52, 41]. More recently, weight-only, activation-aware, and outlier-suppressing quantization methods have extended these ideas to large-scale Transformer models [31, 61, 43, 64, 28]. Across all of these directions, the central challenge is not merely reducing bit-width but controlling the approximation error introduced when continuous-valued parameters and activations are mapped to a discrete representation.

Binarization represents the extreme of this design space, constraining weights, activations, or both to a single bit and replacing floating-point arithmetic with bitwise operations [4, 17], [68, 36]. The resulting efficiency gains are substantial, but so is the loss of representational capacity, far exceeding that of conventional low-bit quantization. Recovering this capacity has driven a large body of work proposing compensating scaling factors [17], multiple binary bases [45], improved residual connections [48], activation reshaping [47], information-preserving binarization functions [54], rotation-based weight alignment [44], adaptive binary sets [19], periodic surrogate gradients [20], and more stable latent-weight optimization [62]. These methods have since been extended to language Transformers [25, 53, 58, 49] and Vision Transformers [6], [65, 39, 7], each requiring substantial architecture-specific redesign to address the distinct failure modes of attention-based models. These methods collectively establish binarization as a viable compression paradigm, but they also expose a persistent limitation: the techniques that recover accuracy tend to be tightly coupled to the architecture for which they were designed, a fragmentation that the present work directly addresses.

The efficiency motivation for binary neural networks is clear: replacing full precision floating-point operations with bitwise operations can dramatically reduce both memory footprint and computational cost. XNOR-Net [17] was among the first to demonstrate this concretely for CNNs, binarizing both weights and activations and introducing scaling factors to partially recover the accuracy lost through extreme quantization. Building on earlier schemes such as BinaryConnect [4] and BinaryNet [36], it established that bitwise convolution was not merely theoretically attractive but practically viable. DoReFa-Net [68] extended this motivation from inference to training, showing that parameter gradients could also be quantized to low-bit values, allowing both forward and backward passes to benefit from low-bit computation. Together, these works defined the core trade-off that would drive the field: efficiency gains come at the cost of representational fidelity, and closing that gap requires understanding precisely where information is lost. Critically, however, each of these methods was designed with convolutional architectures in mind, and the solutions they propose do not transfer cleanly to other settings.

The bulk of subsequent CNN research focused on diagnosing and reducing quantization error from different angles, each proposing fixes that are tightly coupled to the convolutional setting. IR-Net [54] reframed the problem as one of information retention, arguing that binarization should be evaluated not only by forward-pass approximation error but also by how much gradient information is lost during backpropagation. It introduced Libra Parameter Binarization to balance information in the forward pass and the Error Decay Estimator to improve gradient fidelity in the backward pass. RBNN [44] identified a complementary source of error: while most methods compensate for the magnitude difference between full precision and binary weights, they largely ignore angular mismatch between the weight vectors. By introducing a rotation-based binarization that aligns weight directions before applying the sign function, RBNN showed that the geometry of the weight space, not just its scale, matters for accurate binarization. AdaBin [19] pushed this further by questioning whether binary values need to be restricted to the fixed set $\{-1, 1\}$ at all; by learning adaptive real-valued binary sets per layer and using Maxout in place of the standard non-linear activation, it demonstrated that matching the binary representation to the layer's actual distribution could substantially improve network capacity. BiPer [20] took a different approach to the same problem, replacing the sign function in the forward pass with a square-wave periodic function and using a sinusoidal surrogate during the backward pass, combined with a two-stage training procedure that initializes the binary

model from a real-weighted periodic precursor. Despite their differences, all of these methods share a common limitation: their design choices: whether architectural modifications, custom binarization functions, or distribution-matching schemes are specific to convolutional feature extraction and do not generalize naturally to attention-based architectures.

A parallel concern, also largely confined to the CNN literature, is whether binary networks can be reliably optimized at all. ReCU [62] identified the problem of *dead weights*: latent full precision weights that sit in the tails of the weight distribution rarely change sign during training, effectively freezing a portion of the model's representational capacity. The proposed Rectified Clamp Unit draws these weights toward the center of the distribution, increasing the likelihood that they participate meaningfully in optimization. This shift in focus from minimizing quantization error to improving the trainability of the latent weights themselves reflects a maturation in how the field understood binary optimization. Yet even this more principled view of the problem was developed and validated exclusively in convolutional settings, leaving open the question of how optimization instability manifests in Transformer architectures, where the loss landscape and gradient dynamics differ substantially.

These challenges take on new dimensions in Transformer architectures, where performance depends not only on local feature extraction but on self-attention, token interactions, and learned embedding representations. Crucially, the solutions developed for binary CNNs do not transfer: attention mechanisms introduce their own failure modes that require architecture-specific remedies. BinaryBERT [25] was an early attempt to push BERT quantization toward the 1-bit limit, and its central finding was instructive: reducing weights from ternary to binary precision causes a sharp performance drop and an irregular loss landscape. The proposed solution, ternary weight splitting, in which a half-sized ternary model is trained and then equivalently split to initialize a full-sized binary one is a BERT-specific initialization strategy with no obvious analog in convolutional or visual settings. BiBERT [53] tackled the harder setting of fully binarized BERT, where weights, embeddings, and activations are all 1-bit, and identified two bottlenecks specific to this setting: information collapse in binary self-attention, where direct binarization of attention scores weakens the model's ability to distinguish relevant from irrelevant tokens, and optimization direction mismatch during knowledge distillation. The proposed Bi-Attention mechanism and Direction-Matching Distillation scheme address these problems effectively within the BERT setting, but both are designed around the structure of language model attention and do not straightforwardly apply to visual architectures or convolutional ones.

Binary Vision Transformers inherit both the efficiency motivation of binary CNNs and the attention-specific challenges identified in binary BERT, while adding the further requirement of visual representation across spatial patch tokens. Here too, existing methods respond by introducing architecture-specific modifications rather than general principles. ViT-1.58b [65] approaches this by adopting 1.58-bit ternary weights rather than strict binary ones, constraining parameters to $\{-1, 0, 1\}$ while retaining 8-bit activations, and replacing standard linear layers with BitLinear layers tailored to the ViT encoder structure. BHViT [6] targets the more aggressive setting of fully 1-bit weights and activations and proposes a hybrid architecture that uses convolutional token mixing in early stages and attention-based mixing in later stages, complemented by Quantization Decomposition for binary attention, a shift-enhanced binary MLP, and a regularization loss to reduce weight oscillation. While effective, both methods are purpose-built for Vision Transformers and cannot be applied to CNNs or language Transformers without significant redesign.

Taken together, the existing literature reveals a fragmented landscape: binary CNNs, binary language Transformers, and binary Vision Transformers have each developed their own binarization strategies, training procedures, and architectural modifications, with little transfer between settings. This fragmentation is not incidental as it reflects the fact that prior methods treat binarization as an architecture-specific engineering problem rather than a general optimization principle. In contrast, the approach proposed in this work is designed to be architecture-agnostic, applying uniformly to convolutional, language Transformer, and Vision Transformer settings without requiring task-specific modifications. This generality distinguishes it fundamentally from the methods surveyed above and motivates the unified framework developed here.

B Proof of Proposition 2.1

Proposition 2.1. *Let $x_1, \dots, x_n \in \mathbb{S}^{p-1}$, let $w \in \mathbb{R}^p \setminus \{0\}$, and assume that w is not aligned with any x_i, i.e., $\|w\|_2^2 - (w^\top x_i)^2 > 0$, for $i = 1, \dots, n$. Consider evaluating $\tilde{\nabla}\mathcal{L}(w)$ in (3) using a Gaussian embedding matrix $\mathcal{G} \in \mathbb{R}^{N \times p}$. For every $\varepsilon > 0$ and $\delta \in (0, 1)$, with probability at least $1 - \delta$, if*

$$N \geq \frac{32}{\pi^2 \|w\|_2^2 \varepsilon^2} \log\left(\frac{2n}{\delta}\right),$$

then $\|\tilde{\nabla}\mathcal{L}(w) - \nabla\mathcal{L}(w)\|_2 \leq \varepsilon$.

Proof. Consider g_j to be the j-th row of $\mathcal{G}$, for $j = 1, \dots, N$. Explicitly writing the gradient discrepancy yields

$$\begin{aligned}
&\tilde{\nabla}\mathcal{L}(w) - \nabla\mathcal{L}(w) \\
&\quad = -\frac{4}{n\pi} \sum_{i=1}^{n} \left[\frac{1}{N} \mathtt{sign}\left(w^\top \mathcal{G}^\top\right) \mathtt{sign}\left(\mathcal{G} x_i\right) - \frac{2}{\pi} \arcsin\left(\frac{w^\top x_i}{\|w\|_2}\right) \right] \frac{x_i - \frac{w^\top x_i}{\|w\|_2^2} w}{\sqrt{\|w\|_2^2 - (w^\top x_i)^2}} \\
&\quad = -\frac{4}{n\pi} \sum_{i=1}^{n} \left[\frac{1}{N} \sum_{j=1}^{N} \operatorname{sign}(g_j^\top w) \operatorname{sign}(g_j^\top x_i) - \frac{2}{\pi} \arcsin\left(\frac{w^\top x_i}{\|w\|_2}\right) \right] \frac{x_i - \frac{w^\top x_i}{\|w\|_2^2} w}{\sqrt{\|w\|_2^2 - (w^\top x_i)^2}}.
\end{aligned}$$

Notice that since $\|x_i\|_2 = 1$,

$$\left\| x_i - \frac{w^\top x_i}{\|w\|_2^2} w \right\|_2^2 = 1 - \frac{(w^\top x_i)^2}{\|w\|_2^2},$$

which implies

$$\left\| \frac{x_i - \frac{w^\top x_i}{\|w\|_2^2} w}{\sqrt{\|w\|_2^2 - (w^\top x_i)^2}} \right\|_2 = \frac{1}{\|w\|_2}.$$

By Grothendieck's identity we know that

$$\mathbb{E}\left[\operatorname{sign}(g_j^\top w) \operatorname{sign}(g_j^\top x_i)\right] = \frac{2}{\pi} \arcsin\left(\frac{w^\top x_i}{\|w\|_2}\right).$$

On the other hand the main element of the discrepancy vector can be written as

$$\Delta_i = \frac{1}{N} \sum_{j=1}^{N} \operatorname{sign}(g_j^\top w) \operatorname{sign}(g_j^\top x_i) - \frac{2}{\pi} \arcsin\left(\frac{w^\top x_i}{\|w\|_2}\right).$$

Because the summands are bounded in $[-1, 1]$, by Hoeffding's inequality, for every $t > 0$,

$$\mathbb{P}\left(|\Delta_i| > t\right) \leq 2 \exp\left(-\frac{N t^2}{2}\right).$$

Applying a union bound over $i = 1, \dots, n$, we obtain

$$\mathbb{P}\left(\max_{1 \leq i \leq n} |\Delta_i| > t\right) \leq 2n \exp\left(-\frac{N t^2}{2}\right).$$

On the event $\max_i |\Delta_i| \leq t$, we have

$$\|\tilde{\nabla}\mathcal{L}(w) - \nabla\mathcal{L}(w)\|_2 \leq \frac{4}{n\pi} \sum_{i=1}^{n} |\Delta_i| \left\| \frac{x_i - \frac{w^\top x_i}{\|w\|_2^2} w}{\sqrt{\|w\|_2^2 - (w^\top x_i)^2}} \right\|_2 \leq \frac{4}{n\pi} \sum_{i=1}^{n} t \cdot \frac{1}{\|w\|_2} = \frac{4t}{\pi \|w\|_2}.$$

Choosing $t = \pi\|w\|_2\varepsilon/4$, yields $\|\tilde{\nabla}\mathcal{L}(w) - \nabla\mathcal{L}(w)\|_2 \le \varepsilon$. The failure probability is then bounded by

$$2n\exp\left(-\frac{N}{2}\left(\frac{\pi\|w\|_2\varepsilon}{4}\right)^2\right) = 2n\exp\left(-\frac{N\pi^2\|w\|_2^2\varepsilon^2}{32}\right).$$

To make this probability at most δ, it is sufficient that

$$2n\exp\left(-\frac{N\pi^2\|w\|_2^2\varepsilon^2}{32}\right) \le \delta,$$

which is equivalent to

$$N \ge \frac{32}{\pi^2\|w\|_2^2\varepsilon^2}\log\left(\frac{2n}{\delta}\right).$$

□

C Proof of Proposition 2.2

Proposition 2.2 (Recovery from vanishing G-gradient)**.** *Assume that $\Pi_w^\perp x_i \neq 0$ for all $x_i \in \mathbb{S}^{p-1}$, $i = 1,\dots,n$. If $\ker(\mathcal{A}(w)) \cap \{-1,0,1\}^N = \{0\}$, and $\tilde{\nabla}\mathcal{L}(w) = 0$, then $w_b = w_b^\star$.*

Proof. Let $\delta = w_b - w_b^\star \in \{-2,0,2\}^N$. By construction,

$$\tilde{\nabla}\mathcal{L}(w) = \frac{4}{n\pi N}\mathcal{A}(w)\delta.$$

If $\tilde{\nabla}\mathcal{L}(w) = 0$, then $\mathcal{A}(w)\delta = 0$, so $\delta \in \ker(\mathcal{A}(w)) \cap \{-2,0,2\}^N$. By assumption, this intersection is $\{0\}$, hence $\delta = 0$, and therefore $w_b = w_b^\star$. □

D Proof of Proposition 2.3

Proposition 2.3 (Recovery from small G-gradient)**.** *For $x_i \in \mathbb{S}^{p-1}$, assume that $\Pi_w^\perp x_i \neq 0$, and $\lambda_i(w) \ge \alpha_\lambda > 0$, for $i = 1,\dots,n$ and $n \ge 4$. Further for $X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{p\times n}$ assume $X^\top\Pi_w^\perp X \succeq \alpha_\Pi I_n \succ 0$. If $\ker(\mathcal{A}(w)) \cap \{-1,0,1\}^N = \{0\}$, and $\|\tilde{\nabla}\mathcal{L}(w)\|_2 < 16\alpha_\lambda\sqrt{\alpha_\Pi}/(n\pi N)$, then $w_b = w_b^\star$.*

Proof. We proceed by setting $\mathcal{S} = \mathtt{sign}(X^\top G^\top)$, and $\Lambda = \mathrm{diag}(\lambda_1(w),\dots,\lambda_n(w))$, then alternatively the G-gradient can be written as

$$\tilde{\nabla}\mathcal{L}(w) = \frac{4}{n\pi N}\mathcal{A}(w)\delta = \frac{4}{n\pi N}\Pi_w^\perp X\Lambda\mathcal{S}\delta.$$

Since $\Pi_w^\perp$ is a projection matrix, the assumption $X^\top\Pi_w^\perp X \succeq \alpha_\Pi I_n$ implies that

$$\forall z \in \mathbb{R}^n: \qquad \|\Pi_w^\perp Xz\|_2 = \sqrt{z^\top X^\top\Pi_w^{\perp\top}\Pi_w^\perp Xz} = \sqrt{z^\top X^\top\Pi_w^\perp Xz} \ge \sqrt{\alpha_\Pi}\|z\|_2. \tag{7}$$

Let $\delta' = \frac{1}{2}\delta \in \{-1,0,1\}^N$. Applying (7) to $z = \Lambda\mathcal{S}\delta'$ gives

$$\left\|\tilde{\nabla}\mathcal{L}(w)\right\|_2 = \frac{8}{n\pi N}\left\|\Pi_w^\perp X\Lambda\mathcal{S}\delta'\right\|_2 \ge \frac{8\sqrt{\alpha_\Pi}}{n\pi N}\|\Lambda\mathcal{S}\delta'\|_2 \ge \frac{8\alpha_\lambda\sqrt{\alpha_\Pi}}{n\pi N}\|\mathcal{S}\delta'\|_2. \tag{8}$$

We prove the claim by contradiction. Assume $\delta \neq 0$, which, by the null-space property, requires the integer-valued vector $\mathcal{S}\delta'$ to be nonzero. Let $m = |\operatorname{supp}(\delta')|$ denote the number of nonzero elements in δ'. If m is odd, then every coordinate of $\mathcal{S}\delta'$ is an odd integer, and therefore $\|\mathcal{S}\delta'\|_2 \ge \sqrt{n}$. If m

is even, then every coordinate of $\mathcal{S}\delta'$ is even, and since $\mathcal{S}\delta' \neq 0$, we have $\|\mathcal{S}\delta'\|_2 \geq 2$. Since $n \geq 4$, it follows that whenever $\delta \neq 0$, we have $\|\mathcal{S}\delta'\|_2 \geq \min\{\sqrt{n}, 2\} \geq 2$, which, using (8), requires

$$\left\|\tilde{\nabla}\mathcal{L}(w)\right\|_2 \geq \frac{16\alpha_\lambda\sqrt{\alpha_\Pi}}{n\pi N}.$$

This contradicts the proposition's assumption. Therefore, $\delta = 0$, which implies $w_b = w_b^\star$.

□

E Proof of Theorem 2.1

Theorem 2.1. *Assume that $\Pi_w^\perp x_i \neq 0$ for $x_i \in \mathbb{S}^{p-1}$, and let $X = [x_1, \ldots, x_n]$. Assume that $0 \prec \alpha_L I_n \preceq X^\top X \preceq \alpha_U I_n$ and $X^\top \Pi_w^\perp X \succ 0$. Defining $\kappa_X = (2\alpha_U - \alpha_L)/(2 - \alpha_L)$, there exists a universal constant $C > 0$ such that for every $\beta \in (0,1)$, if $n \geq C\kappa_X (N/\log N + \log(N/\beta))$, then*

$$\mathbb{P}\left(\ker(\mathcal{A}(w)) \cap \{-1, 0, 1\}^N = \{0\}\right) \geq 1 - \beta.$$

To establish the result, we first introduce a number of auxiliary lemmas, which are then used to develop the main proof in Subsection E.2.

E.1 Auxiliary Lemmas

Lemma E.1 (Gaussian Brascamp–Lieb Inequality)**.** *Let $(u_1, \ldots, u_n)$ be a centered jointly Gaussian vector, where each $u_i \in \mathbb{R}^m$. Suppose*

$$\mathrm{Cov}(u_1, \ldots, u_n) \preceq \kappa \ \mathrm{diag}\left(\mathrm{Cov}(u_1), \ldots, \mathrm{Cov}(u_n)\right),$$

for some $\kappa \geq 1$. Then, for every collection of nonnegative measurable functions $f_i : \mathbb{R}^m \to [0, \infty)$ we have

$$\mathbb{E}\prod_{i=1}^n f_i(u_i) \leq \prod_{i=1}^n \left(\mathbb{E} f_i(u_i)^\kappa\right)^{1/\kappa}.$$

Proof. This lemma is an implication of a fundamental result by Lieb [42] regarding the maximizers of integral inequalities involving Gaussian kernels. The lemma is a straightforward variation of the following theorem adapted from [27].

Theorem E.2 (Theorem 1 of [27])**.** *Let $n, m_1, \ldots, m_n$ be positive integers and let $N = \sum_{i=1}^n m_i$. Suppose that u_i is a m_i-dimensional random vector for $1 \leq i \leq n$ such that their joint law, $u = (u_1, \ldots, u_n)$, forms a centered jointly N-dimensional Gaussian random vector with covariance matrix $T = (T_{ij})_{1 \leq i,j \leq n}$, where T_{ij} is the covariance matrix between u_i and u_j for $1 \leq i, j \leq n$. Let P be the block diagonal matrix, $P = \mathrm{diag}(p_1 T_{11}, \ldots, p_n T_{nn})$. For any set of nonnegative measurable functions f_i on $\mathbb{R}^{m_i}$, if $T \preceq P$, then*

$$\mathbb{E}\prod_{i=1}^n f_i(u_i) \leq \prod_{i=1}^n \left(\mathbb{E} f_i(u_i)^{p_i}\right)^{1/p_i}.$$

Setting $u = (u_1, \ldots, u_n) \in \mathbb{R}^{nm}$, and picking $p_i = \kappa$ and $m_i = m$ implies Lemma E.1. □

Lemma E.2. *For every integer $m \geq 1$,*

$$\binom{2m}{m} \leq \frac{4^m}{\sqrt{3m+1}} < \frac{4^m}{\sqrt{2m}}.$$

Proof. We prove the claim by induction on m. For $m = 1$, $\binom{2}{1} = 2$ and $\frac{4}{\sqrt{4}} = 2$, so the claim holds with equality. Assume now that the claim holds for some $m \geq 1$, namely

$$\binom{2m}{m} \leq \frac{4^m}{\sqrt{3m+1}}.$$

Using

$$\binom{2m+2}{m+1} = \binom{2m}{m} \frac{(2m+2)(2m+1)}{(m+1)^2} = \binom{2m}{m} \frac{2(2m+1)}{m+1},$$

we obtain

$$\binom{2m+2}{m+1} \leq \frac{4^m}{\sqrt{3m+1}} \frac{2(2m+1)}{m+1}.$$

It remains to show that

$$\frac{4^m}{\sqrt{3m+1}} \frac{2(2m+1)}{m+1} \leq \frac{4^{m+1}}{\sqrt{3m+4}}.$$

After canceling 4^m, and squaring both sides, it suffices to check

$$\frac{(2m+1)^2}{4(m+1)^2} \leq \frac{3m+1}{3m+4}.$$

Cross-multiplying gives $(2m+1)^2(3m+4) \leq 4(m+1)^2(3m+1)$. Indeed,

$$4(m+1)^2(3m+1) - (2m+1)^2(3m+4) = m > 0.$$

Therefore,

$$\binom{2m+2}{m+1} \leq \frac{4^{m+1}}{\sqrt{3m+4}}.$$

This proves the induction step and completes the proof. □

Lemma E.3. *Consider given matrices $X = [x_1, \ldots, x_n] \in \mathbb{R}^{p \times n}$ where $x_i \in \mathbb{S}^{p-1}$, and $Z = [z_1, \ldots z_n] \in \mathbb{R}^{p \times n}$. Let $\mathcal{G} \in \mathbb{R}^{N \times p}$ be a Gaussian matrix. Define $\mathcal{A} = \sum_{i=1}^n z_i \mathtt{sign}(\mathcal{G} x_i)^\top$. Assume that $0 \prec \alpha_L I_n \preceq X^\top X \preceq \alpha_U I_n$ and Z has full column rank. For*

$$\kappa_X = \frac{2\alpha_U - \alpha_L}{2 - \alpha_L},$$

$C > 0$ a universal constant, and $\beta \in (0,1)$, if $n \geq C\kappa_X \left(\frac{N}{\log N} + \log \frac{N}{\beta} \right)$, then

$$\mathbb{P}\left(\ker(\mathcal{A}) \cap \{-1, 0, 1\}^N = \{0\} \right) \geq 1 - \beta.$$

Proof. First, note that $\alpha_L \in (0, 1]$, which guarantees the positivity of κ_X. The reason for that is

$$\alpha_L \leq \lambda_{\min}(X^\top X) \leq \frac{1}{n} \operatorname{Tr}\left(X^\top X \right) = \frac{1}{n} \sum_{i=1}^n \|x_i\|_2^2 = 1.$$

A similar argument yields $\alpha_U \geq 1$, which guarantees that $\kappa_X \geq 1$.

Setting $\mathcal{S} = \mathtt{sign}(X^\top \mathcal{G}^\top)$ allows formulating $\mathcal{A} = Z\mathcal{S}$. Since Z has full column rank, $\mathcal{A}\delta = 0$ if and only if $\mathcal{S}\delta = 0$. Therefore,

$$\mathbb{P}\left(\ker(\mathcal{A}) \cap \{-1, 0, 1\}^N = \{0\} \right) = \mathbb{P}\left(\ker(\mathcal{S}) \cap \{-1, 0, 1\}^N = \{0\} \right).$$

We now fix $\delta \in \{-1, 0, 1\}^N$ and define $m = |\operatorname{supp}(\delta)|$, which represents the number of non-zero entries in δ. In the remainder of the proof, we will bound $\mathbb{P}(\mathcal{S}\delta = 0)$.

Consider $g_j \sim \mathcal{N}(0, I_p)$ to represent the j-th row of $\mathcal{G}$. Defining the Gaussian vectors $\mathcal{z}_j = X^\top g_j$, it naturally follows that $\mathcal{z}_1, \ldots, \mathcal{z}_N$ are independent and

$$\mathcal{z}_j \sim \mathcal{N}(0, X^\top X), \qquad j = 1, \ldots, N.$$

By the construction of $\mathcal{S}$ we have $\mathcal{S}\delta = \sum_{j=1}^{N} \delta_j \mathtt{sign}(\zeta_j)$. Since $\delta_j \in \{-1, 1\}$ on $\mathrm{supp}(\delta)$, and because g_j is centered Gaussian, the random vector $\delta_j g_j$ has the same distribution as g_j. Hence, using $\zeta_j = X^\top g_j$, we have

$$\delta_j \zeta_j = X^\top (\delta_j g_j) \overset{d}{=} X^\top g_j = \zeta_j.$$

Consequently, by the symmetry and independence of the Gaussian vectors ζ_j,

$$\sum_{j \in \mathrm{supp}(\delta)} \delta_j \mathtt{sign}(\zeta_j) \overset{d}{=} \sum_{j=1}^{m} \mathtt{sign}(\zeta_j),$$

where $\zeta_1, \ldots, \zeta_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, X^\top X)$, and $\overset{d}{=}$ denotes following identical distributions. Therefore

$$\mathbb{P}(\mathcal{S}\delta = 0) = \mathbb{P}\left(\sum_{j=1}^{m} \mathtt{sign}(\zeta_j) = 0 \right).$$

Clearly, if m is odd, then each coordinate of $\sum_{j=1}^{m} \mathtt{sign}(\zeta_j)$ is a sum of an odd number of ± 1 entries, and hence cannot equal zero. Therefore

$$m = 2k + 1,\ k \in \mathbb{Z}: \qquad \mathbb{P}(\mathcal{S}\delta = 0) = 0.$$

Now assume m is even, and define $\Sigma = X^\top X - \frac{\alpha_L}{2} I_n$. The well conditioning of $X^\top X$ implies

$$0 \prec \frac{\alpha_L}{2} I_n \preceq \Sigma \preceq \left(\alpha_U - \frac{\alpha_L}{2} \right) I_n.$$

Notice that since the entries on the diagonal of $X^\top X$ are all one, we have $(\Sigma)_{ii} = 1 - \frac{\alpha_L}{2}$, for $i = 1, \ldots, n$. Since $X^\top X = \frac{\alpha_L}{2} I_n + \Sigma$, we can represent ζ_j in distribution as

$$\zeta_j \overset{d}{=} \sqrt{\frac{\alpha_L}{2}}\, \xi_j + \eta_j,$$

where $\xi_j \sim \mathcal{N}(0, I_n)$ and $\eta_j \sim \mathcal{N}(0, \Sigma)$ are independent. Clearly, across different j's, all these random vectors are independent. Consider $\tilde{\xi}_1, \ldots, \tilde{\xi}_m \sim \mathcal{N}(0, 1)$ to be i.i.d. random variables and define

$$q_m(u_1, \ldots, u_m) = \mathbb{P}_{\tilde{\xi}} \left(\sum_{j=1}^{m} \mathrm{sign}\left(\sqrt{\frac{\alpha_L}{2}}\, \tilde{\xi}_j + u_j \right) = 0 \right).$$

Conditioned on $\eta_1, \ldots, \eta_m$, the randomness in different coordinates i comes from independent Gaussian variables $(\xi_j)_i$. Therefore,

$$\mathbb{P}\left(\sum_{j=1}^{m} \mathrm{sign}(\zeta_j) = 0 \,\middle|\, \eta_1, \ldots, \eta_m \right) = \prod_{i=1}^{n} q_m\big((\eta_1)_i, \ldots, (\eta_m)_i\big),$$

where $(\eta_j)_i$ denotes the i-th coordinate of η_j. Taking an expectation gives

$$\mathbb{P}(\mathcal{S}\delta = 0) = \mathbb{E} \prod_{i=1}^{n} q_m\big((\eta_1)_i, \ldots, (\eta_m)_i\big). \tag{9}$$

We now bound the right-hand side of (9) using the Gaussian Brascamp–Lieb Inequality in Lemma E.1. For this purpose, consider defining the random vectors $u_i = \big((\eta_1)_i, \ldots, (\eta_m)_i\big) \in \mathbb{R}^m$, and the joint vector $u = (u_1, \ldots, u_n) \in \mathbb{R}^{mn}$, which is a centered Gaussian. Since the η_j's are independent and each has covariance Σ, the covariance matrix of u is $\Sigma \otimes I_m$. Each vector u_i has covariance $(1 - \alpha_L/2) I_m$. Furthermore,

$$\Sigma \otimes I_m \preceq \left(\alpha_U - \frac{\alpha_L}{2} \right) I_n \otimes I_m = \kappa_X \left(\left(1 - \frac{\alpha_L}{2} \right) I_n \otimes I_m \right).$$

Now appealing to Lemma E.1 guarantees that

$$\mathbb{E} \prod_{i=1}^{n} q_m\big((\eta_1)_i, \ldots, (\eta_m)_i\big) = \mathbb{E} \prod_{i=1}^{n} q_m(u_i) \leq \prod_{i=1}^{n} \left(\mathbb{E} q_m(v)^{\kappa_X} \right)^{1/\kappa_X},$$

where $\mathfrak{v} \sim \mathcal{N}(0, (1-\alpha_L/2)I_m)$. Clearly, $0 \le q_m \le 1$ and as stated at the beginning of the proof, $\kappa_X \ge 1$, which yields $q_m(\mathfrak{v})^{\kappa_X} \le q_m(\mathfrak{v})$. Therefore

$$\mathbb{P}(\mathcal{S}\delta = 0) \le \left(\mathbb{E}q_m(\mathfrak{v})\right)^{n/\kappa_X}.$$

We now compute and bound $\mathbb{E}q_m(\mathfrak{v})$. Since $\mathfrak{v} \sim \mathcal{N}(0, (1-\alpha_L/2)I_m)$, and $\tilde{\xi} \sim \mathcal{N}(0, I_m)$ is independent of $\mathfrak{v}$, we have

$$\sqrt{\frac{\alpha_L}{2}}\,\tilde{\xi} + \mathfrak{v} \sim \mathcal{N}(0, I_m).$$

Consequently, the vector $\mathtt{sign}(\sqrt{\alpha_L/2}\tilde{\xi} + \mathfrak{v})$ contains independent Rademacher random entries and

$$\mathbb{E}q_m(\mathfrak{v}) = \mathbb{P}\left(1^\top \mathtt{sign}(\sqrt{\alpha_L/2}\tilde{\xi} + \mathfrak{v}) = 0\right) = \frac{\binom{m}{m/2}}{2^m} \le m^{-1/2},$$

where the inequality uses Lemma E.2. Therefore, for even m: $\mathbb{P}(\mathcal{S}\delta = 0) \le (m^{-1/2})^{n/\kappa_X} = m^{-n/(2\kappa_X)}$. Summarizing the even and odd cases gives

$$\mathbb{P}(\mathcal{S}\delta = 0) \le \begin{cases} m^{-n/(2\kappa_X)}, & m \text{ even}, \\ 0, & m \text{ odd}. \end{cases} \tag{10}$$

Union Bound Over All Ternary Vectors. We now generalize the argument from a fixed δ to all ternary vectors. Clearly, for a fixed support size m, the number of vectors $\delta \in \{-1,0,1\}^N$ with $|\operatorname{supp}(\delta)| = m$ is $\binom{N}{m}2^m$. Therefore,

$$\begin{aligned}\mathbb{P}\left(\exists \delta \in \{-1,0,1\}^N \setminus \{0\} : \mathcal{S}\delta = 0\right) &\le \sum_{m=1}^{N} \sum_{\substack{\delta \in \{-1,0,1\}^N \\ |\operatorname{supp}(\delta)| = m}} \mathbb{P}(\mathcal{S}\delta = 0) \\ &\le \sum_{\substack{2 \le m \le N \\ m \text{ even}}} \binom{N}{m} 2^m m^{-n/(2\kappa_X)}.\end{aligned}$$

Using the inequality $\binom{N}{m} \le \left(\frac{eN}{m}\right)^m$, we have

$$\binom{N}{m}2^m m^{-n/(2\kappa_X)} \le \exp\left[m \log\left(\frac{2eN}{m}\right) - \frac{n}{2\kappa_X}\log m\right].$$

To acquire the advertised bound, it is enough to require that, for all $2 \le m \le N$,

$$\frac{n}{2\kappa_X}\log m \ge m \log\left(\frac{2eN}{m}\right) + \log\frac{N}{\beta}.$$

Equivalently, it is enough that

$$n \ge 2\kappa_X \sup_{2 \le m \le N} \frac{m\log(2eN/m) + \log(N/\beta)}{\log m}.$$

Since

$$\sup_{2 \le m \le N} \frac{m\log(2eN/m)}{\log m} \le C\frac{N}{\log N}$$

for a universal constant $C > 0$, and since

$$\frac{\log(N/\beta)}{\log m} \le \frac{\log(N/\beta)}{\log 2},$$

a sufficient condition would be $n \ge C\kappa_X\left(N/\log N + \log(N/\beta)\right)$, with a larger universal constant $C > 0$. Under this condition, we will have

$$\sum_{\substack{2 \le m \le N \\ m \text{ even}}} \binom{N}{m} 2^m m^{-n/(2\kappa_X)} \le \beta.$$

In other words, under this sufficient condition, $\mathbb{P}\left(\ker(\mathcal{A}) \cap \{-1,0,1\}^N = \{0\}\right) \ge 1-\beta$, which completes the proof. □

E.2 Main Proof of Theorem 2.1

The theorem's proof is a direct application of Lemma E.3. Setting $\mathcal{S} = \texttt{sign}(X^\top \mathcal{G}^\top)$, and $\Lambda = \operatorname{diag}(\lambda_1(w), \ldots, \lambda_n(w))$, we have

$$\mathcal{A}(w) = \Pi_w^\perp X \Lambda \mathcal{S}.$$

To apply Lemma E.3, it suffices to view $\Pi_w^\perp X \Lambda$ as the Z matrix and argue that it has full column rank. By the theorem's assumption, $X^\top \Pi_w^\perp X \succ 0$. Since $X^\top \Pi_w^\perp X = X^\top \Pi_w^{\perp\top} \Pi_w^\perp X$ and Λ is a diagonal matrix with positive entries, we have $\Lambda^\top X^\top \Pi_w^{\perp\top} \Pi_w^\perp X \Lambda \succ 0$, which makes $\Pi_w^\perp X \Lambda$ full column rank, and Lemma E.3 applicable to $\mathcal{A}(w)$.

F Proof of Proposition 2.4

Proposition 2.4. *Define $\sigma_\delta(t) = \operatorname{erf}(t/(\delta\sqrt{2}))$, where $\operatorname{erf}(t) = 2/\sqrt{\pi}\int_0^t e^{-z^2} dz$ denotes the error function. If $u, v \in \mathbb{R}^p$ are fixed and g is a Gaussian vector, then*

$$\mathbb{E}\left[\sigma_\delta(g^\top u)\ \operatorname{sign}(g^\top v)\right] = \frac{2}{\pi} \arcsin\left(\frac{u^\top v}{\sqrt{\|u\|_2^2 + \delta^2}\,\|v\|_2}\right).$$

.

Proof. Let $\xi \sim \mathcal{N}(0,1)$ be independent of g. We first note that

$$\sigma_\delta(t) = \operatorname{erf}\left(\frac{t}{\delta\sqrt{2}}\right) = 2\phi\left(\frac{t}{\delta}\right) - 1 = \mathbb{E}_\xi[\operatorname{sign}(t + \delta\xi)],$$

where ϕ denotes the standard normal cumulative distribution function. Indeed,

$$\mathbb{E}_\xi[\operatorname{sign}(t+\delta\xi)] = \mathbb{P}(t + \delta\xi \geq 0) - \mathbb{P}(t + \delta\xi < 0) = 2\Phi\left(\frac{t}{\delta}\right) - 1.$$

Hence,

$$\begin{aligned}\mathbb{E}\left[\sigma_\delta(g^\top u)\operatorname{sign}(g^\top v)\right] &= \mathbb{E}_g \mathbb{E}_\xi\left[\operatorname{sign}(g^\top u + \delta\xi)\operatorname{sign}(g^\top v)\,\middle|\, g\right] \\ &= \mathbb{E}_{g,\xi}\left[\operatorname{sign}(g^\top u + \delta\xi)\operatorname{sign}(g^\top v)\right].\end{aligned}$$

Now define the augmented Gaussian vector

$$\widetilde{g} = \begin{bmatrix} g \\ \xi \end{bmatrix} \sim \mathcal{N}(0, I_{p+1}), \qquad \tilde{u} = \begin{bmatrix} u \\ \delta \end{bmatrix}, \qquad \tilde{v} = \begin{bmatrix} v \\ 0 \end{bmatrix}.$$

Then it is clear that

$$\widetilde{g}^\top \tilde{u} = g^\top u + \delta\xi, \qquad \widetilde{g}^\top \tilde{v} = g^\top v,$$

and therefore, $\mathbb{E}\left[\sigma_\delta(g^\top u)\operatorname{sign}(g^\top v)\right] = \mathbb{E}\left[\operatorname{sign}(\widetilde{g}^\top \tilde{u})\operatorname{sign}(\widetilde{g}^\top \tilde{v})\right]$. Applying the Grothendieck identity yields

$$\mathbb{E}\left[\operatorname{sign}(\widetilde{g}^\top \tilde{u})\operatorname{sign}(\widetilde{g}^\top \tilde{v})\right] = \frac{2}{\pi}\arcsin\left(\frac{\tilde{u}^\top \tilde{v}}{\|\tilde{u}\|_2\,\|\tilde{v}\|_2}\right) = \frac{2}{\pi}\arcsin\left(\frac{u^\top v}{\sqrt{\|u\|_2^2 + \delta^2}\,\|v\|_2}\right),$$

which completes the proof. □

G Proof of Theorem 2.2

Theorem 2.2. *Let $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$ be bounded below by $\mathcal{L}^\star \in \mathbb{R}$, and μ-smooth, that is*

$$\forall w, w' \in \mathbb{R}^p : \qquad \mathcal{L}(w') \le \mathcal{L}(w) + \nabla\mathcal{L}(w)^\top (w' - w) + \frac{\mu}{2}\|w' - w\|_2^2. \tag{11}$$

Assume moreover that for all $w \in \mathbb{R}^p$, $g : \mathbb{R}^p \to \mathbb{R}^p$ satisfies $\|\nabla\mathcal{L}(w) - g(w)\|_2 \le \varepsilon$. Consider the iteration $w_{k+1} = w_k - \eta\, g(w_k)$, with stepsize $\eta \in (0, 1/\mu)$. Then $\min_{0\le k\le T-1} \|g(w_k)\|_2^2 \le 2\,(\mathcal{L}(w_0) - \mathcal{L}^\star)/(T(\eta - \mu\eta^2)) + \varepsilon^2/\,(1 - \mu\eta)^2$ for every integer $T \ge 1$. In particular, for $\eta = \frac{1}{2\mu}$,

$$\min_{0\le k\le T-1} \|g(w_k)\|_2^2 \;\le\; \frac{8\mu\,(\mathcal{L}(w_0) - \mathcal{L}^\star)}{T} + 4\varepsilon^2. \tag{12}$$

Proof. Define the approximation error at iteration k by $e_k = \nabla\mathcal{L}(w_k) - g(w_k)$, for which $\|e_k\|_2 \le \varepsilon$ for every k. A direct implication of the smoothness is

$$\begin{aligned}
\mathcal{L}(w_{k+1}) &\le \mathcal{L}(w_k) + \langle \nabla\mathcal{L}(w_k),\, w_{k+1} - w_k\rangle + \mu\|w_{k+1} - w_k\|_2^2 \\
&= \mathcal{L}(w_k) + \big\langle \nabla\mathcal{L}(w_k),\, -\eta\, g(w_k)\big\rangle + \mu\eta^2\|g(w_k)\|_2^2 \\
&= \mathcal{L}(w_k) - \eta\|g(w_k)\|_2^2 - \eta\langle e_k,\, g(w_k)\rangle + \mu\eta^2\|g(w_k)\|_2^2 \\
&\le \mathcal{L}(w_k) - \big(\eta - \mu\eta^2\big)\|g(w_k)\|_2^2 + \eta\varepsilon\,\|g(w_k)\|_2,
\end{aligned} \tag{13}$$

where we used the Cauchy–Schwarz in the last inequality as

$$-\eta\langle e_k,\, g(w_k)\rangle \;\le\; \eta\|e_k\|_2\,\|g(w_k)\|_2 \;\le\; \eta\varepsilon\,\|g(w_k)\|_2.$$

Furthermore, we have

$$\eta\varepsilon\,\|g(w_k)\|_2 \;\le\; \frac{\eta(1 - \mu\eta)}{2}\|g(w_k)\|_2^2 + \frac{\eta\varepsilon^2}{2(1 - \mu\eta)}, \tag{14}$$

which uses the inequality $bt \le \frac{a}{2}t^2 + \frac{b^2}{2a}$, for $a > 0,\ t \ge 0$. Combining (14) with (13) and rearranging gives

$$\frac{\eta(1 - \mu\eta)}{2}\|g(w_k)\|_2^2 \;\le\; \mathcal{L}(w_k) - \mathcal{L}(w_{k+1}) + \frac{\eta\varepsilon^2}{2(1 - \mu\eta)}. \tag{15}$$

Summing (15) from $k = 0$ to $T - 1$ and using telescoping gives

$$\frac{\eta(1 - \mu\eta)}{2}\sum_{k=0}^{T-1}\|g(w_k)\|_2^2 \le \mathcal{L}(w_0) - \mathcal{L}(w_T) + \frac{T\eta\varepsilon^2}{2(1 - \mu\eta)} \le \mathcal{L}(w_0) - \mathcal{L}^\star + \frac{T\eta\varepsilon^2}{2(1 - \mu\eta)}. \tag{16}$$

Dividing (16) by T and using $\min_{0\le k\le T-1}\|g(w_k)\|_2^2 \;\le\; \frac{1}{T}\sum_{k=0}^{T-1}\|g(w_k)\|_2^2$, we obtain

$$\min_{0\le k\le T-1} \|g(w_k)\|_2^2 \;\le\; \frac{2\big(\mathcal{L}(w_0) - \mathcal{L}^\star\big)}{T\eta(1 - \mu\eta)} + \frac{\varepsilon^2}{(1 - \mu\eta)^2}.$$

Finally, through the special choice $\eta = \frac{1}{2\mu}$, we get

$$\min_{0\le k\le T-1} \|g(w_k)\|_2^2 \;\le\; \frac{8\mu\,(\mathcal{L}(w_0) - \mathcal{L}^\star)}{T} + 4\varepsilon^2.$$

□

H Proof of Proposition 2.5

Proposition 2.5. *Consider $\mathcal{L}_\delta(w)$ as stated in (4), with $x_i \in \mathbb{S}^{p-1}$. Then $\mathcal{L}_\delta(w)$ is $\mu_{\mathcal{L}}$-smooth, where*

$$\mu_{\mathcal{L}} = \frac{8}{\pi^2\delta^2} + \frac{16}{n\pi\delta^2}\sum_{i=1}^{n}(|y_i| + 1).$$

Proof. For each i, define

$$\phi_i(w) = \frac{2}{\pi}\arcsin\left(\frac{w^\top x_i}{\sqrt{\|w\|_2^2+\delta^2}}\right), \qquad \ell_i(w) = (y_i - \phi_i(w))^2.$$

Then

$$\mathcal{L}_\delta(w) = \frac{1}{n}\sum_{i=1}^n \ell_i(w).$$

We first derive uniform bounds on $\nabla\phi_i$ and $\nabla^2\phi_i$.

Gradient Bound for ϕ_i. Since $\|x_i\|_2 = 1$, we have

$$|w^\top x_i| \le \|w\|_2 < \sqrt{\|w\|_2^2+\delta^2},$$

so the argument of $\arcsin$ always lies in $(-1,1)$, and ϕ_i is well defined and smooth on all of $\mathbb{R}^p$. Accordingly, a direct differentiation gives

$$\|\nabla\phi_i(w)\|_2 = \frac{2}{\pi}\frac{\|(\|w\|_2^2+\delta^2)x_i - (w^\top x_i)w\|_2}{(\|w\|_2^2+\delta^2)\sqrt{\|w\|_2^2+\delta^2-(w^\top x_i)^2}}.$$

Noting that

$$\begin{aligned}\|(\|w\|_2^2+\delta^2)x_i - (w^\top x_i)w\|_2^2 &= (\|w\|_2^2+\delta^2)\big(\|w\|_2^2+\delta^2-(w^\top x_i)^2\big) - \delta^2(w^\top x_i)^2\\ &\le (\|w\|_2^2+\delta^2)\big(\|w\|_2^2+\delta^2-(w^\top x_i)^2\big).\end{aligned}$$

Therefore,

$$\forall w\in\mathbb{R}^p: \qquad \|\nabla\phi_i(w)\|_2 \le \frac{2}{\pi}\frac{\sqrt{(\|w\|_2^2+\delta^2)\big(\|w\|_2^2+\delta^2-(w^\top x_i)^2\big)}}{(\|w\|_2^2+\delta^2)\sqrt{\|w\|_2^2+\delta^2-(w^\top x_i)^2}} = \frac{2}{\pi\sqrt{\|w\|_2^2+\delta^2}} \le \frac{2}{\pi\delta}. \tag{17}$$

Hessian Bound for ϕ_i. We fix i. By rotational invariance, we may work in coordinates where $x_i = e_1$. Since orthogonal changes of variables preserve operator norms of Hessians, it suffices to bound the Hessian in these coordinates. Rewriting

$$w = \begin{bmatrix}u\\ v\end{bmatrix}, \qquad u\in\mathbb{R}, \quad v\in\mathbb{R}^{p-1},$$

gives $w^\top x_i = u$, and $\|w\|_2^2 = u^2+\|v\|_2^2$, which implies

$$\phi_i(w) = \frac{2}{\pi}\arcsin\left(\frac{u}{\sqrt{u^2+\|v\|_2^2+\delta^2}}\right) = \frac{2}{\pi}\arctan\left(\frac{u}{\sqrt{\|v\|_2^2+\delta^2}}\right),$$

where we used the identity $\arcsin\Big(\frac{a}{\sqrt{a^2+b^2}}\Big) = \arctan\big(\frac{a}{b}\big)$ for $b>0$. Thus it suffices to bound the Hessian of

$$h(u,v) = \arctan\left(\frac{u}{\sqrt{\|v\|_2^2+\delta^2}}\right).$$

Notice that the gradient components follow the forms

$$\partial_u h(u,v) = \frac{\sqrt{\|v\|_2^2+\delta^2}}{u^2+\|v\|_2^2+\delta^2}, \qquad \nabla_v h(u,v) = -\frac{u}{\sqrt{\|v\|_2^2+\delta^2}\,\big(u^2+\|v\|_2^2+\delta^2\big)}\,v.$$

For the second derivative with respect to u,

$$\partial_{uu}h(u,v) = \frac{\partial}{\partial u}\left(\frac{\sqrt{\|v\|_2^2+\delta^2}}{u^2+\|v\|_2^2+\delta^2}\right) = -\frac{2u\sqrt{\|v\|_2^2+\delta^2}}{\big(u^2+\|v\|_2^2+\delta^2\big)^2}.$$

Hence

$$|\partial_{uu} h(u,v)| = \frac{2|u|\sqrt{\|v\|_2^2+\delta^2}}{\left(u^2+\|v\|_2^2+\delta^2\right)^2} \le \frac{1}{u^2+\|v\|_2^2+\delta^2} \le \frac{1}{\delta^2}, \tag{18}$$

where we used the fact that $2|u|\sqrt{\|v\|_2^2+\delta^2} \le u^2+\|v\|_2^2+\delta^2$. Next, for the mixed derivative, we have

$$\begin{aligned}\nabla_v \partial_u h(u,v) &= \frac{\left(u^2+\|v\|_2^2+\delta^2\right)\frac{v}{\sqrt{\|v\|_2^2+\delta^2}} - \sqrt{\|v\|_2^2+\delta^2}\,(2v)}{\left(u^2+\|v\|_2^2+\delta^2\right)^2} \\ &= \frac{u^2-\|v\|_2^2-\delta^2}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)^2}\, v.\end{aligned}$$

Therefore,

$$\begin{aligned}\|\nabla_v \partial_u h(u,v)\|_2 &\le \frac{u^2+\|v\|_2^2+\delta^2}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)^2}\,\|v\|_2 \\ &= \frac{\|v\|_2}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)} \\ &\le \frac{1}{\delta^2}.\end{aligned} \tag{19}$$

Finally, for the final Hessian block, using the product rule we have

$$\nabla_v^2 h(u,v) = -\frac{u}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)} I_{p-1} + v\big(\nabla_v a(u,v)\big)^\top,$$

where

$$a(u,v) = -\frac{u}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)}.$$

Differentiating $a(u,v)$ gives

$$\nabla_v a(u,v) = \frac{u\big(u^2+3(\|v\|_2^2+\delta^2)\big)}{(\|v\|_2^2+\delta^2)^{3/2}\big(u^2+\|v\|_2^2+\delta^2\big)^2}\, v.$$

Hence

$$\nabla_v^2 h(u,v) = -\frac{u}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)} I_{p-1} + \frac{u\big(u^2+3(\|v\|_2^2+\delta^2)\big)}{(\|v\|_2^2+\delta^2)^{3/2}\big(u^2+\|v\|_2^2+\delta^2\big)^2}\, vv^\top.$$

We now bound the operator norm of this block. Since $\|vv^\top\| = \|v\|_2^2 \le \|v\|_2^2+\delta^2$,

$$\begin{aligned}\|\nabla_v^2 h(u,v)\| \le{}& \frac{|u|}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)} \\ &+ \frac{|u|\big(u^2+3(\|v\|_2^2+\delta^2)\big)}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)^2}.\end{aligned}$$

Again, using $2|u|\sqrt{\|v\|_2^2+\delta^2} \le u^2+\|v\|_2^2+\delta^2$, we get

$$\frac{|u|}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)} \le \frac{1}{2(\|v\|_2^2+\delta^2)} \le \frac{1}{2\delta^2},$$

and similarly

$$\frac{|u|\big(u^2+3(\|v\|_2^2+\delta^2)\big)}{\sqrt{\|v\|_2^2+\delta^2}\left(u^2+\|v\|_2^2+\delta^2\right)^2} \le \frac{3}{2(\|v\|_2^2+\delta^2)} \le \frac{3}{2\delta^2},$$

which together imply

$$\|\nabla_v^2 h(u,v)\| \le \frac{2}{\delta^2}. \tag{20}$$

Collecting the block estimates in (18), (19) and (20) bounds the Hessian of h as

$$\|\nabla^2 h(u,v)\| \le \frac{4}{\delta^2}.$$

Finally, since $\phi_i = (2/\pi)h$, we conclude that

$$\forall w \in \mathbb{R}^p : \qquad \|\nabla^2 \phi_i(w)\| \le \frac{2}{\pi} \cdot \frac{4}{\delta^2} = \frac{8}{\pi\delta^2}.$$

Combining the Bounds. Since $\ell_i(w) = (y_i - \phi_i(w))^2$, we have

$$\nabla \ell_i(w) = -2(y_i - \phi_i(w))\,\nabla \phi_i(w),$$

and

$$\nabla^2 \ell_i(w) = 2\nabla\phi_i(w)\nabla\phi_i(w)^\top - 2(y_i - \phi_i(w))\,\nabla^2\phi_i(w).$$

Taking operator norms and using $|\phi_i(w)| \le 1$, we obtain

$$\begin{aligned}
\|\nabla^2 \ell_i(w)\| &\le 2\|\nabla\phi_i(w)\|^2 + 2|y_i - \phi_i(w)|\,\|\nabla^2\phi_i(w)\| \\
&\le 2\left(\frac{2}{\pi\delta}\right)^2 + 2(|y_i|+1)\left(\frac{8}{\pi\delta^2}\right) \\
&= \frac{8}{\pi^2\delta^2} + \frac{16(|y_i|+1)}{\pi\delta^2}.
\end{aligned}$$

Finally by the triangle inequality we get

$$\begin{aligned}
\|\nabla^2 \mathcal{L}_\delta(w)\| &\le \frac{1}{n}\sum_{i=1}^n \|\nabla^2 \ell_i(w)\| \\
&\le \frac{1}{n}\sum_{i=1}^n \left(\frac{8}{\pi^2\delta^2} + \frac{16(|y_i|+1)}{\pi\delta^2}\right) \\
&= \frac{8}{\pi^2\delta^2} + \frac{16}{n\pi\delta^2}\sum_{i=1}^n (|y_i|+1),
\end{aligned}$$

which completes the proof. □

I Proof of Theorem 2.3

Theorem 2.3. *Consider $\mathcal{L}_\delta(w)$ and $g_\delta(w)$ as stated in (4) and (5), where $x_i \in \mathbb{S}^{p-1}$, and $\mathcal{G} \in \mathbb{R}^{N\times p}$ is a Gaussian matrix. Then, there exists a universal constant $C > 0$ such that for every $\beta \in (0,1)$:*

$$\sup_{w\in\mathbb{R}^p} \|\nabla\mathcal{L}_\delta(w) - g_\delta(w)\|_2 \le \frac{C}{\delta}\left(\sqrt{\frac{p}{N}} + \frac{p}{N} + \sqrt{\frac{\log(2n/\beta)}{N}}\right), \tag{21}$$

with probability at least $1-\beta$.

To prove the result, we first state a set of auxiliary lemmas, which are then used as tools in the main proof presented in Subsection I.2.

I.1 Auxiliary Lemmas

To prove Theorem 2.3, we first need to state some auxiliary lemmas as presented below.

Lemma I.1. *Let $x \in \mathbb{S}^{p-1}$ and $\delta > 0$. Then, for every $w \in \mathbb{R}^p$,*

$$\frac{\left\|x - \frac{w^\top x}{\|w\|_2^2+\delta^2} w\right\|_2}{\sqrt{(\|w\|_2^2+\delta^2) - (w^\top x)^2}} \le \frac{1}{\sqrt{\|w\|_2^2+\delta^2}} \le \frac{1}{\delta}.$$

Proof. Notice that since $x \in \mathbb{S}^{p-1}$, we have $\|x\|_2 = 1$. Expanding the squared norm in the numerator gives

$$\left\| x - \frac{w^\top x}{\|w\|_2^2 + \delta^2} w \right\|_2^2 = 1 - \frac{2(w^\top x)^2}{\|w\|_2^2 + \delta^2} + \frac{(w^\top x)^2 \|w\|_2^2}{(\|w\|_2^2 + \delta^2)^2}.$$

Combining the last two terms gives

$$\left\| x - \frac{w^\top x}{\|w\|_2^2 + \delta^2} w \right\|_2^2 = 1 - \frac{(\|w\|_2^2 + 2\delta^2)(w^\top x)^2}{(\|w\|_2^2 + \delta^2)^2} \leq 1 - \frac{(w^\top x)^2}{\|w\|_2^2 + \delta^2} = \frac{(\|w\|_2^2 + \delta^2) - (w^\top x)^2}{\|w\|_2^2 + \delta^2}.$$

Dividing the two sides by $(\|w\|_2^2 + \delta^2) - (w^\top x)^2$ and taking square roots yields

$$\frac{\left\| x - \frac{w^\top x}{\|w\|_2^2 + \delta^2} w \right\|_2}{\sqrt{(\|w\|_2^2 + \delta^2) - (w^\top x)^2}} \leq \frac{1}{\sqrt{\|w\|_2^2 + \delta^2}} \leq \frac{1}{\delta}.$$

□

Recall that a VC-subgraph class is a real-valued function class whose subgraphs have finite VC complexity, a property that leads to polynomial covering-number bounds. In our setting, this framework allows us to control the complexity of the function class generated by $w \mapsto \sigma_\delta(w^\top g)\,\mathrm{sign}(x^\top g)$, which is the key step for obtaining empirical-process bounds uniformly over $w \in \mathbb{R}^p$. The following two lemmas are standard VC-subgraph results adapted from [38], with the statements rewritten in terms of VC-dimension rather than VC-index.

Lemma I.2 (Lemma 9.6 in [38]). *If $\mathcal{F}$ is a finite-dimensional vector space of measurable real-valued functions, then $\mathcal{F}$ is VC-subgraph and*

$$\mathrm{d}_{VC}(\mathcal{F}) \leq \dim(F) + 1.$$

Lemma I.3 (Lemma 9.9 in [38]). *Let $\mathcal{F}$ be a VC-subgraph class with VC-dimension $\mathrm{d}_{VC}(\mathcal{F})$. Then:*

1. *if g is a fixed measurable function, then $F \cdot g = \{fg : f \in \mathcal{F}\}$ is VC-subgraph with VC-dimension at most $2\,\mathrm{d}_{VC}(\mathcal{F})$;*
2. *if $\phi : \mathbb{R} \to \mathbb{R}$ is monotone, then $\phi \circ \mathcal{F} = \{\phi \circ f : f \in \mathcal{F}\}$ is VC-subgraph and $\mathrm{d}_{VC}(\phi \circ \mathcal{F}) \leq \mathrm{d}_{VC}(\mathcal{F})$.*

We also present the following results from [55]. Here, $\mathcal{N}_c(\varepsilon, \mathcal{F}, L_2(\mathcal{Q}))$ denotes the $L_2(\mathcal{Q})$ covering number of the function class $\mathcal{F}$, the minimum number of $L_2(\mathcal{Q})$-balls of radius ε needed to cover $\mathcal{F}$.

Lemma I.4 (Theorem 7.12 of [55]). *Let $\mathcal{F}$ be a VC class of measurable functions with measurable envelope function F_{env}. Then for every $r \geq 1$, every probability measure $\mathcal{Q}$ with $\|F_{\mathrm{env}}\|_{\mathcal{Q},r} > 0$, and every $0 < \varepsilon < 1$,*

$$\mathcal{N}_c(\varepsilon \|F_{\mathrm{env}}\|_{\mathcal{Q},r}, \mathcal{F}, L_r(\mathcal{Q})) \leq K\ \mathrm{d}_{VC}(\mathcal{F})\,(4e)^{\mathrm{d}_{VC}(\mathcal{F})} \left(\frac{2}{\varepsilon}\right)^{r\,\mathrm{d}_{VC}(\mathcal{F})},$$

where K is a universal constant.

The next result converts a polynomial covering-number bound into a bound on the expected supremum of the centered empirical process. This will later allow us to pass from the entropy estimate established above to a quantitative control of the approximation error.

Lemma I.5 (Theorem 7.13 of [55]). *Let $\mathcal{F}$ be a measurable class of functions with constant envelope U, and let $x_1, \dots, x_N$ be i.i.d. random variables with common law $\mathcal{P}$. Assume that for some $A > e^2$, $V \geq 2$, and every finitely supported probability measure $\mathcal{Q}$,*

$$\mathcal{N}_c(\varepsilon U, \mathcal{F}, L_2(\mathcal{Q})) \leq \left(\frac{A}{\varepsilon}\right)^V, \qquad 0 \leq \varepsilon < 1.$$

Then

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}\left|\sum_{j=1}^{N}\big(f(x_j) - \mathbb{E}[f(x_j)]\big)\right|\right] \leq L\left(\sqrt{N\sigma^2 V \log(AU/\sigma)} \;\vee\; VU\log(AU/\sigma)\right),$$

where L is a universal constant and σ satisfies $\sup_{f\in\mathcal{F}} \mathrm{Var}(f(X_1)) \leq \sigma^2$.

Finally, to upgrade the expectation bound obtained above to a high-probability bound, we will use McDiarmid's bounded-differences inequality. The bounded-differences property implies that changing a single input coordinate, while keeping all others fixed, can alter the value of the function by only a controlled amount.

Lemma I.6 (McDiarmid's Bounded-Differences Inequality, see [50, 26]). *Suppose $\mathfrak{z} = f(x_1, \dots, x_N)$ where x_i are independent random variables and f satisfies the bounded-differences property with constants $c_1, \dots, c_N$, namely changing the i-th coordinate while holding the others fixed changes the value of f by at most c_i. Then for all $t \geq 0$,*

$$\mathbb{P}(|\mathfrak{z} - \mathbb{E}_{\mathfrak{z}}| > t) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^N c_i^2}\right).$$

I.2 Main Proof of Theorem 2.3

Proof. Throughout the proof, $\delta > 0$ is fixed, $x_i \in \mathbb{S}^{p-1}$, $y_i \in \mathbb{R}$, and $\mathcal{G} \in \mathbb{R}^{N\times p}$ has i.i.d. Gaussian rows $g_1, \dots, g_N \sim \mathcal{N}(0, I_p)$, which immediately implies that

$$\frac{1}{N}\mathtt{s}_\delta(w^\top \mathcal{G}^\top)\mathtt{sign}(\mathcal{G}x) = \frac{1}{N}\sum_{j=1}^{N}\sigma_\delta(w^\top g_j)\,\mathrm{sign}(g_j^\top x).$$

Also, for a given function f we use the following notations

$$\mathscr{P}_N f := \frac{1}{N}\sum_{j=1}^{N} f(g_j), \qquad \mathcal{P}f := \mathbb{E}[f(g)],$$

where $g \sim \mathcal{N}(0, I_p)$. To impose only minimal assumptions on the data samples $x_i \in \mathbb{S}^{p-1}$, we first analyze the discrepancy in the single-sample setting and then extend the argument to the n-sample case by a union bound. In this regard, for a fixed $x \in \mathbb{S}^{p-1}$ and $y \in \mathbb{R}$, define

$$\hat{\mathcal{L}}_\delta(w) = \left(y - \frac{2}{\pi}\arcsin\left(\frac{w^\top x}{\sqrt{\|w\|_2^2 + \delta^2}}\right)\right)^2,$$

and

$$\hat{g}_\delta(w) = -\frac{4}{\pi}\frac{\left(y - \frac{1}{N}\mathtt{s}_\delta(w^\top\mathcal{G}^\top)\mathtt{sign}(\mathcal{G}x)\right)}{\sqrt{(\|w\|_2^2+\delta^2) - (w^\top x)^2}}\left(x - \frac{w^\top x}{\|w\|_2^2+\delta^2}\,w\right).$$

The exact gradient of $\hat{\mathcal{L}}_\delta(w)$ can be derived as

$$\nabla\hat{\mathcal{L}}_\delta(w) = -\frac{4}{\pi}\frac{\left(y - \frac{2}{\pi}\arcsin\left(\frac{w^\top x}{\sqrt{\|w\|_2^2+\delta^2}}\right)\right)}{\sqrt{(\|w\|_2^2+\delta^2) - (w^\top x)^2}}\left(x - \frac{w^\top x}{\|w\|_2^2+\delta^2}w\right).$$

By proposition 2.4, the numerators expressions in $\hat{g}_\delta(w)$ and $\nabla\hat{\mathcal{L}}_\delta(w)$ can be related via

$$\mathbb{E}\big[\sigma_\delta(w^\top g)\,\mathrm{sign}(g^\top x)\big] = \frac{2}{\pi}\arcsin\left(\frac{w^\top x}{\sqrt{\|w\|_2^2+\delta^2}}\right).$$

We proceed by defining a function class $\mathcal{F}$ as a collection of measurable real-valued functions indexed by $w \in \mathbb{R}^p$:

$$\mathcal{F} = \{f_w : w \in \mathbb{R}^p\}, \qquad f_w(g) = \sigma_\delta(w^\top g)\,\mathrm{sign}(x^\top g).$$

This allows us to write the difference between $\nabla\hat{\mathcal{L}}_\delta(w)$ and $\hat{g}_\delta(w)$ as

$$\nabla\hat{\mathcal{L}}_\delta(w) - \hat{g}_\delta(w) = \frac{4}{\pi}\big((\mathcal{P}_N - \mathcal{P})f_w\big)\frac{x - \frac{w^\top x}{\|w\|_2^2+\delta^2}w}{\sqrt{(\|w\|_2^2+\delta^2)-(w^\top x)^2}}. \tag{22}$$

Combining (22) with Lemma I.1, we obtain

$$\sup_{w\in\mathbb{R}^p}\|\nabla\hat{\mathcal{L}}_\delta(w) - \hat{g}_\delta(w)\|_2 \le \frac{4}{\pi\delta}\sup_{w\in\mathbb{R}^p}|(\mathcal{P}_N - \mathcal{P})f_w|. \tag{23}$$

Thus our problem reduces to bounding the empirical process

$$\mathcal{E}_N := \sup_{w\in\mathbb{R}^p}|(\mathcal{P}_N - \mathcal{P})f_w| = \sup_{f\in\mathcal{F}}|(\mathcal{P}_N - \mathcal{P})f|.$$

Bounding the Empirical Process. Using the aforementioned lemmas we can show that $\mathcal{F}$ is a bounded VC-subgraph class. To do this, consider first the linear class

$$\mathcal{H} = \{h_w : g \mapsto w^\top g,\ w \in \mathbb{R}^p\}.$$

This is a p-dimensional vector space of measurable functions, so by Lemma I.2, $\mathrm{d}_{VC}(\mathcal{H}) \le p+1$. Since σ_δ is monotonically increasing, part 2 of Lemma I.3 implies that $\mathrm{d}_{VC}(\sigma_\delta \circ \mathcal{H}) \le p+1$. Considering the fixed measurable function $s(g) = \mathrm{sign}(x^\top g)$, one can view our target class as $\mathcal{F} = (\sigma_\delta \circ \mathcal{H})\cdot s$, the complexity which can be bounded by appealing again to Lemma I.3 (part 1):

$$\mathrm{d}_{VC}(\mathcal{F}) \le 2(p+1). \tag{24}$$

Moreover, regarding the envelope of $\mathcal{F}$, since $|\sigma_\delta| \le 1$, every $f \in \mathcal{F}$ satisfies $|f(g)| \le 1$ for all $f \in \mathcal{F},\ g \in \mathbb{R}^p$. Using these observations in Lemma I.4 with $r = 2$ and envelope $F_{\mathrm{env}} \equiv 1$, yields the entropy bound

$$\mathcal{N}_c(\varepsilon, \mathcal{F}, L_2(\mathcal{Q})) \le K\,(2p+2)\,(4e)^{2p+2}\left(\frac{2}{\varepsilon}\right)^{2(2p+2)}, \qquad 0<\varepsilon<1,$$

for every probability measure $\mathcal{Q}$. Hence there exist universal constants $A_0 > e^2$ and $V_0 = 4(p+1)$ such that

$$\mathcal{N}_c(\varepsilon, \mathcal{F}, L_2(\mathcal{Q})) \le \left(\frac{A_0}{\varepsilon}\right)^{V_0}, \qquad 0<\varepsilon<1. \tag{25}$$

We next verify the assumptions of Lemma I.5. First, the covering number bound in (25) holds for every probability measure $\mathcal{Q}$, and therefore in particular for every finitely supported probability measure $\mathcal{Q}$, as required by the lemma. Second, since $|f(g)| \le 1$ for every $f \in \mathcal{F}$, we have

$$\mathcal{P}(f-\mathcal{P}f)^2 = \mathrm{Var}(f(g)) \le \mathbb{E}[f(g)^2] \le 1,$$

and hence $\sup_{f\in\mathcal{F}}\mathcal{P}(f-\mathcal{P}f)^2 \le 1$. Thus we may take $\sigma \le 1$ in Lemma I.5. Applying the lemma yields

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}\left|\sum_{j=1}^N (f(g_j) - \mathcal{P}f)\right|\right] \le L\left(\sqrt{N\,V_0\log(A_0)} \ \vee\ V_0\log(A_0)\right).$$

Dividing by N and using the elementary inequality $a \vee b \le a+b$, we obtain

$$\mathbb{E}[\mathcal{E}_N] \le L\left(\sqrt{\frac{V_0\log(A_0)}{N}} + \frac{V_0\log(A_0)}{N}\right).$$

Finally, since $V_0 = 4p + 4$ and A_0 is a universal constant, all fixed numerical factors can be absorbed into a universal constant $C_1 > 0$, giving

$$\mathbb{E}[\mathcal{E}_N] \le C_1 \left(\sqrt{\frac{p}{N}} + \frac{p}{N} \right).$$

To convert the expectation bound on $\mathcal{E}_N$ into a high-probability bound, we apply Lemma I.6 to

$$\mathcal{E}_N(g_1, \dots, g_N) = \sup_{f \in \mathcal{F}} \left| \frac{1}{N} \sum_{j=1}^{N} f(g_j) - \mathcal{P}f \right|.$$

We now show that $\mathcal{E}_N(g_1, \dots, g_N)$ satisfies the bounded-differences property. To this end, fix $i \in \{1, \dots, N\}$, and suppose that only the ith sample is changed, replacing g_i by another point g_i'. Denote $g = (g_1, \dots, g_i, \dots, g_N)$, and $g^{(i)} = (g_1, \dots, g_i', \dots, g_N)$. Then

$$\left| \mathcal{E}_N(g) - \mathcal{E}_N(g^{(i)}) \right| = \left| \sup_{f \in \mathcal{F}} \left| \frac{1}{N} \sum_{j=1}^{N} f(g_j) - \mathcal{P}f \right| - \sup_{f \in \mathcal{F}} \left| \frac{1}{N} \sum_{j=1}^{N} f(g_j^{(i)}) - \mathcal{P}f \right| \right|.$$

Using the elementary inequality $\left| \sup_{f \in \mathcal{F}} A_f - \sup_{f \in \mathcal{F}} B_f \right| \le \sup_{f \in \mathcal{F}} |A_f - B_f|$, we obtain

$$\begin{aligned} \left| \mathcal{E}_N(g) - \mathcal{E}_N(g^{(i)}) \right| &\le \sup_{f \in \mathcal{F}} \left| \left| \frac{1}{N} \sum_{j=1}^{N} f(g_j) - \mathcal{P}f \right| - \left| \frac{1}{N} \sum_{j=1}^{N} f(g_j^{(i)}) - \mathcal{P}f \right| \right| \\ &\le \sup_{f \in \mathcal{F}} \left| \left(\frac{1}{N} \sum_{j=1}^{N} f(g_j) - \mathcal{P}f \right) - \left(\frac{1}{N} \sum_{j=1}^{N} f(g_j^{(i)}) - \mathcal{P}f \right) \right|, \end{aligned}$$

where for the second inequality we used $||a| - |b|| \le |a - b|$. Since the two samples differ only in the i-th coordinate, all terms cancel except the i-th one, and therefore

$$\left| \mathcal{E}_N(g) - \mathcal{E}_N(g^{(i)}) \right| \le \sup_{f \in \mathcal{F}} \frac{|f(g_i) - f(g_i')|}{N} \le \frac{2}{N},$$

where we used the fact that $|f(g)| \le 1$. Thus $\mathcal{E}_N(g_1, \dots, g_N)$ satisfies the bounded-differences property with constants $c_i = \frac{2}{N}$. We may therefore apply Lemma I.6, which gives

$$\mathbb{P}\big(|\mathcal{E}_N - \mathbb{E}\mathcal{E}_N| > t \big) \le 2 \exp\left(-\frac{2t^2}{\sum_{i=1}^{N} c_i^2} \right) = 2 \exp\left(-\frac{Nt^2}{2} \right).$$

Choosing $t = \sqrt{2 \log(2/\beta)/N}$ yields that, with probability at least $1 - \beta$,

$$\mathcal{E}_N \le \mathbb{E}\mathcal{E}_N + \sqrt{\frac{2 \log(2/\beta)}{N}} \le C_2 \left(\sqrt{\frac{p}{N}} + \frac{p}{N} + \sqrt{\frac{\log(2/\beta)}{N}} \right),$$

for some universal constant $C_2 > 0$.

Extension to Multiple Samples. To extend the result to n samples, observe that the full loss and proxy gradient can be written as averages of their one-sample counterparts:

$$\mathcal{L}_\delta(w) = \frac{1}{n} \sum_{i=1}^{n} \hat{\mathcal{L}}_{\delta,i}(w), \qquad g_\delta(w) = \frac{1}{n} \sum_{i=1}^{n} \hat{g}_{\delta,i}(w),$$

where $\hat{\mathcal{L}}_{\delta,i}$ and $\hat{g}_{\delta,i}$ denote the one-sample loss and proxy obtained from the preceding construction with (x, y) replaced by (x_i, y_i). Hence, $\nabla \mathcal{L}_\delta(w) - g_\delta(w) = \frac{1}{n} \sum_{i=1}^{n} \big(\nabla \hat{\mathcal{L}}_{\delta,i}(w) - \hat{g}_{\delta,i}(w) \big)$, and by the triangle inequality,

$$\sup_{w \in \mathbb{R}^p} \|\nabla \mathcal{L}_\delta(w) - g_\delta(w)\|_2 \le \frac{1}{n} \sum_{i=1}^{n} \sup_{w \in \mathbb{R}^p} \|\nabla \hat{\mathcal{L}}_{\delta,i}(w) - \hat{g}_{\delta,i}(w)\|_2.$$

The bound stated above for $\mathcal{E}_N$ combined with (23) guarantees that for each sample $i \in \{1, \dots, n\}$, with probability at least $1 - \beta/n$,

$$\sup_{w \in \mathbb{R}^p} \|\nabla \hat{\mathcal{L}}_{\delta,i}(w) - \hat{g}_{\delta,i}(w)\|_2 \le \frac{C}{\delta}\left(\sqrt{\frac{p}{N}} + \frac{p}{N} + \sqrt{\frac{\log(2n/\beta)}{N}}\right).$$

By the union bound, with probability at least $1 - \beta$, the above estimate holds simultaneously for all $i = 1, \dots, n$. Therefore, on this event,

$$\begin{aligned}\sup_{w \in \mathbb{R}^p} \|\nabla \mathcal{L}_\delta(w) - g_\delta(w)\|_2 &\le \frac{1}{n}\sum_{i=1}^{n} \frac{C}{\delta}\left(\sqrt{\frac{p}{N}} + \frac{p}{N} + \sqrt{\frac{\log(2n/\beta)}{N}}\right) \\ &= \frac{C}{\delta}\left(\sqrt{\frac{p}{N}} + \frac{p}{N} + \sqrt{\frac{\log(2n/\beta)}{N}}\right),\end{aligned}$$

which completes the proof. □

J Natural Generalization Beyond Binary Systems

A main advantage of the proposed framework is its natural extension beyond binary systems. As an example, we consider the ternary setting, where the basic memory elements are *trits* rather than bits. For a fixed threshold $\alpha > 0$, define

$$\mathrm{tsign}_\alpha(t) = \mathrm{sign}(t)\, 1_{\{|t|>\alpha\}},$$

so that $\mathrm{tsign}_\alpha(t) = 0$ for $|t| \le \alpha$, $\mathrm{tsign}_\alpha(t) = 1$ for $t > \alpha$, and $\mathrm{tsign}_\alpha(t) = -1$ for $t < -\alpha$. Thus, tsign_α is a thresholded generalization of the usual sign function.

Grothendieck's identity admits a natural ternary analogue, and more generally extends to multi-level quantization rules. In the ternary setting, the expected correlation between random ternary embeddings remains an explicit smooth function of the underlying Gaussian correlation parameter, with the arcsine law replaced by a bivariate Gaussian cumulative distribution function (CDF) expression. Since our backpropagation procedure relies only on derivatives of this correlation law, below we also report the corresponding derivative identities.

Proposition J.1 (Ternary Grothendieck's Identity)**.** *Let $g \sim \mathcal{N}(0, I_p)$, fix nonzero $u, v \in \mathbb{R}^p$, and set $\rho = u^\top v/\|u\|_2\|v\|_2$. Then*

$$\begin{aligned}\mathbb{E}\big[\mathrm{tsign}_\alpha(g^\top u)\,\mathrm{tsign}_\alpha(g^\top v)\big] =& \Phi(\alpha_u, \alpha_v; \rho) + \Phi(-\alpha_u, -\alpha_v; \rho) \\ &+ \Phi(\alpha_u, -\alpha_v; \rho) + \Phi(-\alpha_u, \alpha_v; \rho) - 1,\end{aligned}$$

where $\alpha_u = \alpha/\|u\|_2$, $\alpha_v = \alpha/\|v\|_2$, and $\Phi(\cdot, \cdot; \rho)$, denotes the standard bivariate normal CDF with correlation ρ. Moreover, $\frac{\partial}{\partial \rho}\mathbb{E}\big[\mathrm{tsign}_\alpha(g^\top u)\,\mathrm{tsign}_\alpha(g^\top v)\big] = 2\,\phi(\alpha_u, \alpha_v; \rho) + 2\,\phi(\alpha_u, -\alpha_v; \rho)$, where

$$\phi(a, b; \rho) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{a^2 - 2\rho ab + b^2}{2(1-\rho^2)}\right)$$

is the corresponding bivariate normal density.

Proof. Set $x = g^\top u$ and $y = g^\top v$. Then (x, y) is jointly Gaussian with mean zero, variances $\mathrm{Var}(x) = \|u\|_2^2$, $\mathrm{Var}(y) = \|v\|_2^2$ and correlation

$$\mathrm{Corr}(x, y) = \frac{u^\top v}{\|u\|_2\|v\|_2} = \rho.$$

By definition,

$$\mathrm{tsign}_\alpha(x) = 1_{\{x \ge \alpha\}} - 1_{\{x \le -\alpha\}}, \qquad \mathrm{tsign}_\alpha(y) = 1_{\{y \ge \alpha\}} - 1_{\{y \le -\alpha\}},$$

which implies,

$$\begin{aligned}\mathbb{E}[\mathrm{tsign}_\alpha(x)\,\mathrm{tsign}_\alpha(y)] = \mathbb{P}(x \ge \alpha, y \ge \alpha) + \mathbb{P}(x \le -\alpha, y \le -\alpha) \\ - \mathbb{P}(x \ge \alpha, y \le -\alpha) - \mathbb{P}(x \le -\alpha, y \ge \alpha).\end{aligned}$$

Since $x/\|u\|_2$ and $y/\|v\|_2$ form a standard bivariate normal pair with correlation ρ, we have

$$\begin{aligned}\mathbb{P}(x \ge \alpha, y \ge \alpha) &= 1 - \Phi(\alpha_u, \infty; \rho) - \Phi(\infty, \alpha_v; \rho) + \Phi(\alpha_u, \alpha_v; \rho), \\ \mathbb{P}(x \le -\alpha, y \le -\alpha) &= \Phi(-\alpha_u, -\alpha_v; \rho), \\ \mathbb{P}(x \ge \alpha, y \le -\alpha) &= \Phi(\infty, -\alpha_v; \rho) - \Phi(\alpha_u, -\alpha_v; \rho), \\ \mathbb{P}(x \le -\alpha, y \ge \alpha) &= \Phi(-\alpha_u, \infty; \rho) - \Phi(-\alpha_u, \alpha_v; \rho).\end{aligned}$$

where $\alpha_u = \alpha/\|u\|_2$ and $\alpha_v = \alpha/\|v\|_2$. Substituting these identities and using $\Phi(-t, \infty; \rho) = 1 - \Phi(t, \infty; \rho)$, and $\Phi(\infty, -t; \rho) = 1 - \Phi(\infty, t; \rho)$, yields the first result

$$\mathbb{E}\big[\mathrm{tsign}_\alpha(g^\top u)\,\mathrm{tsign}_\alpha(g^\top v)\big] = \Phi(\alpha_u, \alpha_v; \rho) + \Phi(-\alpha_u, -\alpha_v; \rho) + \Phi(\alpha_u, -\alpha_v; \rho) + \Phi(-\alpha_u, \alpha_v; \rho) - 1.$$

A standard differentiation gives

$$\frac{\partial}{\partial \rho}\mathbb{E}[\mathrm{tsign}_\alpha(x)\,\mathrm{tsign}_\alpha(y)] = \phi(\alpha_u, \alpha_v; \rho) + \phi(-\alpha_u, -\alpha_v; \rho) + \phi(\alpha_u, -\alpha_v; \rho) + \phi(-\alpha_u, \alpha_v; \rho),$$

which by symmetry of the Gaussian density: $\phi(-\alpha_u, -\alpha_v; \rho) = \phi(\alpha_u, \alpha_v; \rho)$ and $\phi(-\alpha_u, \alpha_v; \rho) = \phi(\alpha_u, -\alpha_v; \rho)$, delivers the advertised result. □

K Grothendieck's Identity Beyond Gaussian Embeddings

In this section we summarize several facts about random sign embeddings used to approximate angular inner products. Grothendieck's identity is first extended from Gaussian vectors to arbitrary rotationally invariant distributions. This immediately permits Haar-orthogonal embeddings obtained from the polar factor of a Gaussian matrix; within orthogonal blocks, the estimator remains unbiased and has strictly smaller variance than its i.i.d. Gaussian counterpart. We then recall the approximate Rademacher identity of Aghasi et al. [1] and use it to establish a compact expectation theorem for a full-width, one-pass randomized Walsh–Hadamard embedding.

K.1 Gaussian Grothendieck's Identity

Let $u, v \in \mathbb{R}^p$ be nonzero and let $g \sim \mathcal{N}(0, I_p)$. The random-hyperplane form of Grothendieck's identity is

$$\mathbb{E}\big[\mathrm{sign}(g^\top u)\,\mathrm{sign}(g^\top v)\big] = \frac{2}{\pi}\arcsin\left(\frac{u^\top v}{\|u\|_2\,\|v\|_2}\right). \tag{26}$$

Consequently, if $W \in \mathbb{R}^{n\times p}$ has unit-norm rows, $x \in \mathbb{R}^p$ is a unit vector, and $\mathcal{G} \in \mathbb{R}^{N\times p}$ has i.i.d. standard Gaussian entries, then, entrywise,

$$\mathbb{E}\left[\frac{1}{N}\mathtt{sign}(W\mathcal{G}^\top)\mathtt{sign}(\mathcal{G}x)\right] = \frac{2}{\pi}\mathtt{arcsin}(Wx). \tag{27}$$

K.2 Rotational Invariance and Orthogonal Embeddings

Gaussianity is not essential in Eq. (26); only the random direction matters.

Theorem K.1 (Rotationally invariant Grothendieck identity)**.** *Let $a \in \mathbb{R}^p$ be rotationally invariant, meaning $Ua \overset{d}{=} a$ for every orthogonal matrix U, and assume $\mathbb{P}(a = 0) = 0$. Then, for all nonzero $u, v \in \mathbb{R}^p$,*

$$\mathbb{E}\big[\mathrm{sign}(a^\top u)\,\mathrm{sign}(a^\top v)\big] = \frac{2}{\pi}\arcsin\left(\frac{u^\top v}{\|u\|_2\,\|v\|_2}\right). \tag{28}$$

Proof. Write $a = R\theta$, where $R = \|a\|_2 > 0$ and θ is uniform on the unit sphere. The positive radius does not affect either sign. If α is the angle between u and v, rotational invariance reduces the calculation to their two-dimensional span. The two signs disagree on wedges of total angular proportion α/π. Hence the expectation is $1 - 2\alpha/\pi = (2/\pi)\arcsin(\cos\alpha)$. □

This result allows embedding designs with better concentration than the general Gaussian matrices. For instance, orthonormal random embeddings can offer a better concentration around the mean. In particular, let $\mathcal{G} \in \mathbb{R}^{N\times p}$ be standard Gaussian, with $N \geq p$, and define its polar factor

$$\mathcal{Q} = \mathcal{G}(\mathcal{G}^\top\mathcal{G})^{-1/2} \in \mathbb{R}^{N\times p}. \tag{29}$$

Then $\mathcal{Q}^\top\mathcal{Q} = I_p$ and $\mathcal{Q}$ is Haar distributed on the Stiefel manifold. Each row is marginally rotationally invariant, and therefore

$$\mathbb{E}\left[\frac{1}{N}\texttt{sign}(W\mathcal{Q}^\top)\texttt{sign}(\mathcal{Q}x)\right] = \frac{2}{\pi}\texttt{arcsin}(Wx). \tag{30}$$

A strict finite-sample variance comparison follows directly for orthogonal blocks. Suppose $N = \nu p$ and form an $N \times p$ embedding by stacking ν independent square polar factors, each obtained from an independent $p \times p$ Gaussian matrix through Eq. (29). For fixed unit vectors u, v, let $\rho = u^\top v$. The variance of the estimator based on N independent Gaussian hyperplanes is

$$\mathrm{Var}\left[\frac{1}{N}\sum_{j=1}^{N}\mathrm{sign}(g_j^\top u)\,\mathrm{sign}(g_j^\top v)\right] = \frac{1}{N}\left[1 - \left(\frac{2}{\pi}\arcsin\rho\right)^2\right]. \tag{31}$$

Within one orthogonal block, if write $Z_j = \mathrm{sign}(q_j^\top u)\,\mathrm{sign}(q_j^\top v)$, then exchangeability gives

$$\mathrm{Var}\left[\frac{1}{p}\sum_{j=1}^{p}Z_j\right] = \frac{1}{p}\left[1 - \left(\frac{2}{\pi}\arcsin\rho\right)^2\right] + \frac{p-1}{p}\,\mathrm{Cov}(Z_1, Z_2). \tag{32}$$

The orthogonal-hyperplane variance theorem of Ji et al. [9] shows that $\mathrm{Cov}(Z_1, Z_2) < 0$ whenever $-1 < \rho < 1$. Averaging ν independent blocks therefore yields

$$\mathrm{Var}\left[\frac{1}{N}\texttt{sign}(W_i\mathcal{Q}^\top)\texttt{sign}(\mathcal{Q}x)\right] < \frac{1}{N}\left[1 - \left(\frac{2}{\pi}\arcsin(W_ix)\right)^2\right], \tag{33}$$

which is exactly the variance of the corresponding estimator formed from an i.i.d. Gaussian matrix in $\mathbb{R}^{N\times p}$. Here $\mathcal{Q}$ denotes the stacked orthogonal-block embedding; the single tall polar factor in Eq. (29) retains the exact expectation in Eq. (30).

K.3 Rademacher Embeddings

The close behavior of Gaussian and Rademacher projections is quantified by the approximate Grothendieck identity proved in the G-Net paper of Aghasi et al. [1]. Let $\mathfrak{r} \in \{\pm 1\}^p$ have independent Rademacher entries. If u, v are unit vectors satisfying

$$\|u\|_\infty, \|v\|_\infty \leq \frac{C_0}{\sqrt{p}}, \qquad |u^\top v| \leq 1 - \delta, \tag{34}$$

for fixed $C_0, \delta > 0$, then

$$\mathbb{E}\big[\mathrm{sign}(\mathfrak{r}^\top u)\,\mathrm{sign}(\mathfrak{r}^\top v)\big] = \frac{2}{\pi}\arcsin(u^\top v) + \mathcal{O}_{C_0,\delta}(p^{-1/2}). \tag{35}$$

Thus, if $\mathcal{R} \in \{\pm 1\}^{N\times p}$ is an i.i.d. Rademacher matrix and the rows of $W \in \mathbb{R}^{n\times p}$ and x satisfy Eq. (34), then

$$\mathbb{E}\left[\frac{1}{N}\texttt{sign}(W\mathcal{R}^\top)\texttt{sign}(\mathcal{R}x)\right] = \frac{2}{\pi}\texttt{arcsin}(Wx) + \mathcal{O}_{C_0,\delta}(p^{-1/2}) \tag{36}$$

entrywise. This result explains why dense Rademacher embeddings can closely track dense Gaussian embeddings when the input and weight vectors are sufficiently dispersed.

K.4 Full-Width Randomized Walsh–Hadamard Embeddings

The preceding Rademacher approximation motivates structured matrices that retain sign randomness while admitting fast multiplication. Let p be a power of two and let H_p be the normalized symmetric Walsh–Hadamard matrix. Suppose $N = \nu p$. For $b = 1, \dots, \nu$, let

$$\mathcal{D}_b = \mathrm{diag}(\delta_{b1}, \dots, \delta_{bp}), \qquad \delta_{bk} \overset{\text{i.i.d.}}{\sim} \mathrm{Rad}, \tag{37}$$

and define the structured random matrix

$$\mathcal{A} = \begin{bmatrix} H_p \mathcal{D}_1 \\ \vdots \\ H_p \mathcal{D}_\nu \end{bmatrix} \in \mathbb{R}^{N \times p}. \tag{38}$$

Each block implements the one-pass transform $z^\top \mapsto z^\top \mathcal{D}_b H_p$.

Theorem K.2 (Full-width one-pass Hadamard expectation)**.** *Let $W \in \mathbb{R}^{n \times p}$ have unit rows and let $x \in \mathbb{S}^{p-1}$ be unit norm. If the dispersion and nondegeneracy conditions in Eq. (34) hold for x and every row of W, then*

$$\mathbb{E}\left[\frac{1}{N} \mathtt{sign}(W\mathcal{A}^\top)\mathtt{sign}(\mathcal{A}x)\right] = \frac{2}{\pi}\mathtt{arcsin}(Wx) + \mathcal{O}_{C_0,\delta}(p^{-1/2}) \tag{39}$$

entrywise. The case $N = p$ is the single full-width, one-pass formula.

Proof. Let h_j be the j-th column of H_p. For every fixed b and j, $\sqrt{p}\,\mathcal{D}_b h_j$ is an i.i.d. Rademacher vector. Hence, for each row W_i,

$$\mathbb{E}\left[\frac{1}{N}\sum_{b=1}^{\nu}\sum_{j=1}^{p} \mathtt{sign}(W_i \mathcal{D}_b h_j)\mathtt{sign}(h_j^\top \mathcal{D}_b x)\right] = \mathbb{E}_\tau\left[\mathrm{sign}(\tau^\top W_i^\top)\,\mathrm{sign}(\tau^\top x)\right].$$

Applying Eq. (35) proves the claim. Independence among the coordinates within a Hadamard block is not needed for the expectation. □

Each block uses one random sign diagonal and one fast Walsh–Hadamard transform, requiring $O(p \log p)$ arithmetic per vector. The construction is therefore a structured $N \times p$ analogue of the dense Rademacher embedding with the same population expectation up to the $O(p^{-1/2})$ invariance error.

L Experimental Details and Additional Results

This appendix provides the full experimental record supporting the main results. We include architecture and optimization details, expanded result tables, and additional diagnostic analyses. Throughout, we use the notation $(\nu_w/\nu_a) \times \nu$, where ν_w and ν_a denote weight and activation bit-widths. For GGM, ν is the projection expansion ratio: if p is the input dimension of a layer, the random projection dimension is $N = \nu p$. For conventional STE scaling experiments, ν denotes width expansion before binarization, since direct STE layers do not have an explicit projection dimension analogous to N.

L.1 Layer Replacement Policy

GGM is applied as a drop-in replacement for the dominant learned linear transformations in each architecture. In convolutional networks, this includes convolutional and fully connected transformations according to the corresponding binary-network setup. In Transformer-style models, GGM replaces the query, key, value, output, and feed-forward projections unless otherwise stated. Components such as embedding layers, classifier heads, normalization layers, residual additions, and attention softmax are kept full precision unless explicitly noted. For controlled STE baselines, the same set of layers is replaced by the corresponding STE-based quantized layers, ensuring that comparisons isolate the low-bit layer mechanism rather than differences in architecture.

L.2 Controlled CIFAR-10 ViT Study

The controlled CIFAR-10 ViT study in Section 4 compares layer-level binarization mechanisms under a fixed architecture and training recipe. The goal is not to reproduce full method-specific pipelines, but to isolate the effect of direct STE quantization, XNOR-style scaling, learned scaling, adaptive binary values, periodic binarization, and GGM.

Architecture. All methods use the same compact ViT backbone. CIFAR-10 images of size 32×32 are split into 4×4 patches, yielding 64 patch tokens plus one class token. The Transformer has embedding dimension $D = 192$, 12 Transformer blocks, and 3 attention heads. Each block uses RMSNorm before attention and before the MLP, residual connections around both sublayers, cosine-normalized Q, K attention with a learned per-head temperature, and an odd activation gate in the MLP. The MLP expansion ratio is 4, so the hidden dimension is $4D = 768$.

Quantized Layers. The patch embedding convolution, class token, positional embedding, normalization layers, softmax attention, residual connections, and final classifier are kept in full precision. In each Transformer block, the attention projections Q, K, V, the attention output projection, and the two MLP linear layers are replaced. For the FP reference, these modules are ordinary full precision linear layers. For GGM experiments, they are replaced by angular layers. For controlled STE experiments, they are replaced by the corresponding STE-backbone variants given in Table 6. Thus, all compared methods modify exactly the same set of linear operators.

Training. All models are trained from scratch for 450 epochs using AdamW with batch size 512, learning rate 5×10^{-4}, and weight decay 0. We use a OneCycle learning rate schedule with a short warmup phase over the first 15% of training, followed by cosine annealing to a small final learning rate. Exponential moving average (EMA) is not used. Training uses random horizontal flips, random resized crops with scale range $[0.8, 1.0]$, RandAugment with two operations and magnitude 9, and input normalization with mean and standard deviation 0.5 per channel. MixUp or CutMix is applied at the batch level with probability 0.5 and $\alpha = 1.0$. Validation and test use only tensor conversion and the same input normalization.

Controlled STE Preprocessing. The controlled STE layer optionally supports activation centering/std-normalization and weight centering/std-normalization before quantization. In Table 6, we report the best preprocessing setting found for each controlled STE variant.

Table 6: Controlled CIFAR-10 ViT binarization study. All STE baselines use the same ViT backbone, optimizer, and training budget as GGM, and all use clipped STE. GGM uses projected-coordinate quantization with learnable output gain; the 1.58-bit setting denotes ternary quantization, since $\log_2 3 \approx 1.58$.

Method	Quantizer / Scale	Config.	Top-1
ViT (FP)	FP (Full Precision)	32/32	92.49
STE-AMAX	$s\,\mathrm{sgn}(z),\ s = \max\lvert z\rvert$	$(1/1)\times 1.0$	57.52
STE-AMAX+G	$s\,\mathrm{sgn}(z),\ s = \max\lvert z\rvert$	$(1/1)\times 1.0$	58.64
XNOR+G	$s\,\mathrm{sgn}(z),\ s = \mathrm{mean}\lvert z\rvert$	$(1/1)\times 1.0$	78.96
LSCALE+G	$s\,\mathrm{sgn}(z),\ s = \lambda\,\mathrm{mean}\lvert z\rvert$	$(1/1)\times 1.0$	78.96
ADABIN+G	$\beta + s\,\mathrm{sgn}(z-\beta)$	$(1/1)\times 1.0$	79.39
BIPER-LF+G	$s\,\mathrm{sgn}(\sin(\omega z))$	$(1/1)\times 1.0$	77.12
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1)\times 1.0$	**86.51**
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1)\times 1.5$	**88.49**
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1)\times 2.0$	**89.46**
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1)\times 2.5$	**90.09**
GGM (Ours)	RF kernel, $s\,\mathrm{sgn}(z)$	$(1/1)\times 3.0$	**91.31**
GGM (Ours)	RF kernel, $\mathcal{Q}_{\log_2 3}(z) \in \{-s, 0, +s\}$	$(\log_2 3/\log_2 3)\times 1.0$	**90.59**
GGM (Ours)	RF kernel, $\mathcal{Q}_{\log_2 3}(z) \in \{-s, 0, +s\}$	$(\log_2 3/\log_2 3)\times 1.5$	**91.14**
GGM (Ours)	RF kernel, $\mathcal{Q}_{\log_2 3}(z) \in \{-s, 0, +s\}$	$(\log_2 3/\log_2 3)\times 2.0$	**91.92**

L.3 Image Classification

We evaluate GGM on CIFAR-10, TinyImageNet, and ImageNet-1K using convolutional backbones. Across these experiments, GGM replaces the dominant learned transformations while keeping the

Table 7: Top-1 accuracy on CIFAR-10 convolutional models. The symbol * with BiPer emphasizes that its a two-stage training procedure. For VGG-16, baseline results were reproduced from the authors' public implementations because the corresponding VGG-16 numbers were not consistently reported in the original papers. GGM results are reported as averages over 5 runs; for VGG-16, reproduced baseline results are also reported as averages over 5 runs. Standard deviations are omitted because they were not reported for the baseline methods in the original papers.

Network	Method	Config.	Top-1
ResNet-18	FP	32/32	94.80
	XNOR-Net	$(1/1) \times 1.0$	90.50
	IR-Net	$(1/1) \times 1.0$	91.50
	RBNN	$(1/1) \times 1.0$	92.20
	ReCU	$(1/1) \times 1.0$	92.80
	AdaBin	$(1/1) \times 1.0$	93.10
	BiPer*	$(1/1) \times 1.0$	93.75
	GGM (Ours)	$(1/1) \times 1.0$	**93.58**
	GGM (Ours)	$(1/1) \times 2.5$	**94.51**
ResNet-20	FP	32/32	92.10
	IR-Net	$(1/1) \times 1.0$	86.50
	RBNN	$(1/1) \times 1.0$	87.80
	ReCU	$(1/1) \times 1.0$	87.40
	AdaBin	$(1/1) \times 1.0$	88.20
	BiPer*	$(1/1) \times 1.0$	87.50
	GGM (Ours)	$(1/1) \times 1.0$	**88.51**
	GGM (Ours)	$(1/1) \times 2.5$	**89.74**
VGG-16	FP	32/32	94.04
	IR-Net	$(1/1) \times 1.0$	81.85
	RBNN	$(1/1) \times 1.0$	88.13
	AdaBin	$(1/1) \times 1.0$	90.26
	BiPer*	$(1/1) \times 1.0$	91.95
	GGM (Ours)	$(1/1) \times 1.0$	**91.47**
	GGM (Ours)	$(1/1) \times 2.5$	**93.12**

overall training pipeline aligned with the corresponding full precision or binary baselines. We report all GGM configurations using $(\nu_w/\nu_a) \times \nu$.

L.3.1 CIFAR-10 Convolutional Models

For CIFAR-10 convolutional experiments, GGM binarizes all hidden layers, including downsampling layers, while keeping the stem convolution and classification head full precision. All GGM models are trained for 300 epochs using AdamW with a OneCycle learning rate schedule. The maximum learning rate is 3×10^{-3}, with batch size 256, weight decay 0.02, and label smoothing 0.05. Mixed-precision training is enabled throughout. We use a seeded random 10% split of the training set for validation and select the best checkpoint based on validation accuracy.

For baseline results, we report numbers from the corresponding papers whenever available. The exception is VGG-16: the VGG-16 baseline results for the compared methods were not consistently reported in the original papers. We therefore reproduced these baselines by cloning the authors' public GitHub repositories and training the models using their provided implementations and experimental settings, without modifying the baseline code. The VGG-16 baseline results in Table 7 are therefore obtained from our own runs of the original implementations.

Some prior baselines use additional training machinery beyond a single-stage binary training run. In particular, BiPer uses a two-stage procedure: the first stage trains with real-valued weights and binary features, while the second stage initializes from this model and trains with binarized weights. GGM, in contrast, is trained directly in a standard single-stage pipeline. We mark BiPer with * to indicate this additional recipe. Other baselines also introduce specialized binarization mechanisms, such as IR-Net's balanced/standardized binarization and error-decay estimator, ReCU's rectified clamp unit, and AdaBin's adaptive binary sets.

GGM achieves the strongest binary performance across the evaluated CIFAR-10 convolutional architectures. On ResNet-18, GGM reaches 94.51%, exceeding BiPer by 0.76 points and remaining

Table 8: Top-1 accuracy on TinyImageNet. * denotes BiPer, which uses a two stage training procedure. GGM improves consistently as the projection expansion ratio ν increases.

Network	Method	Config.	Top-1
ResNet-18	FP	32/32	65.48
	XNOR-Net	$(1/1) \times 1.0$	48.01
	IR-Net	$(1/1) \times 1.0$	56.12
	RBNN	$(1/1) \times 1.0$	55.79
	AdaBin	$(1/1) \times 1.0$	53.72
	BiPer*	$(1/1) \times 1.0$	57.10
	BiPer	$(1/1) \times 1.0$	45.54
	GGM (Ours)	$(1/1) \times 1.0$	**59.84**
	GGM (Ours)	$(1/1) \times 2.0$	**63.48**
	GGM (Ours)	$(1/1) \times 3.0$	**64.10**
	GGM (Ours)	$(1.58/1.58) \times 1.0$	**63.75**
ResNet-34	FP	32/32	67.85
	IR-Net	$(1/1) \times 1.0$	60.20
	RBNN	$(1/1) \times 1.0$	59.57
	AdaBin	$(1/1) \times 1.0$	54.62
	BiPer*	$(1/1) \times 1.0$	61.41
	BiPer	$(1/1) \times 1.0$	46.20
	GGM (Ours)	$(1/1) \times 1.0$	**64.46**
	GGM (Ours)	$(1/1) \times 2.0$	**66.54**
	GGM (Ours)	$(1/1) \times 3.0$	**67.08**
	GGM (Ours)	$(1.58/1.58) \times 1.0$	**66.93**

0.29 points below full precision. On ResNet-20, GGM reaches 89.74%, improving over AdaBin by 1.54 points. On VGG-16, GGM reaches 93.12%, outperforming BiPer by 1.17 points and reducing the gap to full precision to 0.92 points.

L.3.2 TinyImageNet Convolutional Models

For TinyImageNet, we evaluate GGM on ResNet-18 and ResNet-34 at an input resolution of 64×64. To prevent aggressive downsampling at this lower resolution, we replace the 7×7 convolution in the stem with a 3×3 convolution and remove the max-pooling layer. We keep the stem, the final classification head and all batch-normalization layers in full precision; all remaining convolutional layers are binarized using GGM.

The models are trained for 90 epochs using distributed training across 4 GPUs with a per-GPU batch size of 128 (yielding an effective batch size of 512). We use the AdamW optimizer with a base learning rate of 1×10^{-3}, a weight decay of 0.05, label smoothing of 0.1, dropout of 0.1 and mixed-precision training. Furthermore, we employ Mixup ($\alpha = 0.8$) and CutMix ($\alpha = 1.0$) data augmentations during training. The learning rate schedule incorporates a linear warmup followed by cosine annealing. We use ReLU non-linearity for these experiments except for BiPer that performs better using HardTanh and AdaBin that uses Maxout. In Table 8, we report results across different projection expansion ratio configurations, $\nu \in \{1.0, 2.0, 3.0\}$. Since the baseline papers did not consistently report results on TinyImageNet, we reproduced the results by cloning the author's official public repositories and trained the baseline models following their implementation and training instructions. BiPer uses a two stage training mechanism. We train each stage for 90 epochs (180 in total). For a fair comparison with our single stage training approach, we also merge both BiPer stages and train the fully-binarized model for 200 epochs. * indicates BiPer with two-stage training. GGM achieves the highest accuracy among the evaluated single-stage binarization methods on both architectures. At $\nu = 1$, GGM outperforms IR-Net, the strongest evaluated single-stage baseline, by 3.72 percentage points on ResNet-18 and 4.26 percentage points on ResNet-34.

On both architectures, accuracy improves monotonically with ν. ResNet-18 improves from 59.84% at $(1/1) \times 1.0$ to 63.48% at $(1/1) \times 2.0$, reducing the gap to full precision to 2.0 points. Similarly, ResNet-34 improves from 64.46% to 66.54%, reducing the gap to full precision to 1.31 points. These results show that increasing the random-projection capacity remains effective on TinyImageNet, a harder setting between CIFAR-10 and ImageNet-1K.

Table 9: Top-1 accuracy on ImageNet-1K convolutional models. * denotes BiPer, which uses a two-stage training procedure.

Network	Method	Config.	Top-1
ResNet-18	FP	32/32	69.60
	XNOR-Net	$(1/1) \times 1.0$	51.20
	IR-Net	$(1/1) \times 1.0$	58.10
	RBNN	$(1/1) \times 1.0$	59.90
	ReCU	$(1/1) \times 1.0$	61.00
	AdaBin	$(1/1) \times 1.0$	63.10
	BiPer*	$(1/1) \times 1.0$	61.40
	GGM (Ours)	$(1/1) \times 1.0$	**63.53**
	GGM (Ours)	$(1/1) \times 2.0$	**66.39**
	GGM (Ours)	$(1/1) \times 3.0$	**67.42**
ResNet-34	FP	32/32	73.30
	IR-Net	$(1/1) \times 1.0$	62.90
	RBNN	$(1/1) \times 1.0$	63.10
	ReCU	$(1/1) \times 1.0$	65.10
	AdaBin	$(1/1) \times 1.0$	66.40
	BiPer*	$(1/1) \times 1.0$	65.73
	GGM (Ours)	$(1/1) \times 1.0$	**68.82**
	GGM (Ours)	$(1/1) \times 2.0$	**70.92**
	GGM (Ours)	$(1/1) \times 3.0$	**71.54**

L.3.3 ImageNet-1K Convolutional Models

For ImageNet-1K, we evaluate GGM on ResNet-18 and ResNet-34 at input resolution 224×224. We keep the stem convolution, downsampling layers, and classification head full precision, and binarize the remaining layers with GGM; the downsampling layers are kept full precision to align with common binary-network evaluation protocols. Models are trained for 200 epochs using AdamW with maximum learning rate 3×10^{-3}, weight decay 0, label smoothing 0.1, and mixed precision. We used one H100 gpu with batch size 256. The activation function is replaced with PReLU. The learning rate follows a OneCycle schedule with cosine annealing and is stepped at every optimizer update.

On ImageNet-1K, GGM improves consistently as ν increases, reaching performances closer to full-precision mark. On ResNet-18, GGM reaches 63.53%, 66.39%, and 67.42%, at $\nu = 1.0, 2.0, 3.0$, respectively. The worst ResNet-18 configuration at $\nu = 1.0$ improves over AdaBin by 0.43 points and the best configuration at $\nu = 3.0$ reduces the gap to full precision to 2.18 points. On ResNet-34, GGM reaches 68.82%, 70.92%, and 71.54% at $\nu = 1.0, 2.0, 3.0$, respectively. The improvement over AdaBin ranges from 2.42 to 5.14 percentage points, while the best configuration reduces the gap to full precision to 1.76 points.

L.3.4 CIFAR-10 Vision Transformers

We further contextualize the controlled CIFAR-10 ViT study by comparing GGM with representative prior ViT quantization methods in Table 10. The GGM rows use exactly the same compact ViT backbone, layer replacement policy, optimizer, data augmentation, and training budget described in Section L.2: the patch embedding, class token, positional embedding, normalization layers, attention softmax, residual connections, and classifier remain full precision, while the Q, K, V, attention output, and MLP projections are replaced by GGM layers. Thus, the GGM entries are directly tied to the controlled experiment above.

The prior methods are included as broader reference points rather than as controlled layer-isolation baselines. Where available, we report their results on the same compact CIFAR-10 ViT architecture used in Section L.2; otherwise, we use the closest available CIFAR-10 ViT result from the corresponding paper or public implementation. These methods follow different approaches to quantization. BiT employs robust binarized Transformer training with multi-stage distillation [11], while ViT-1.58b restricts weights to ternary values through a specialized quantization procedure [65]. LSQ learns quantization step sizes during training [5], XNOR-Net uses binary weights and activations with scaling factors [17], and FQ-ViT applies post-training quantization through calibration of a pretrained full-precision ViT [46]. Accordingly, these results should be interpreted as empirical comparisons

Table 10: CIFAR-10 ViT comparison to prior quantization methods. * denotes multi-stage training and ** post-training quantization. GGM rows use the exact controlled architecture and training setup from Section L.2; prior rows use the corresponding CIFAR-10 ViT results from the cited methods or public implementations.

Method	Config	Top-1
Full Precision	32/32	92.49
ViT-1.58b	$(1/1) \times 1.0$	77.77
	$(1.58/1.58) \times 1.0$	81.63
LSQ	$(4/4) \times 1.0$	85.89
XNOR	$(1/1) \times 1.0$	69.20
BiT*	$(1/1) \times 1.0$	82.35
FQ-ViT** (PTQ)	$(4/4) \times 1.0$	84.74
GGM (Ours)	$(1/1) \times 1.0$	**86.51**
	$(1/1) \times 2.0$	**89.46**
	$(1/1) \times 2.5$	**90.09**
	$(1.58/1.58) \times 1.0$	**90.59**

with existing quantization approaches, recognizing that the methods differ in their training objectives, additional stages, precision levels, and calibration requirements.

GGM substantially outperforms all prior binary ViT methods in this comparison. At $(1/1) \times 2.5$, GGM reaches 90.09%, improving over BiT by 7.74 points despite BiT using multi-stage distillation, and exceeding XNOR-Net by 20.89 points. Even without projection expansion beyond $\nu = 1$, GGM reaches 86.51%, already outperforming the prior binary and ternary ViT baselines listed in the table.

GGM also compares favorably to higher-bit methods. The ternary configuration $(1.58/1.58) \times 1.0$ reaches 90.59%, outperforming both 4-bit LSQ and 4-bit FQ-ViT while using lower nominal precision. This supports the main observation from the controlled study, that the angular correlation structure of GGM uses low-bit capacity more effectively than conventional Euclidean quantization schemes, even when those schemes rely on learned step sizes, specialized quantizers, distillation, or post-training calibration.

L.4 TinyImageNet vision Transformer Experiments

Dataset and Shared Architecture. TinyImageNet contains 200 classes, 100,000 training images, and 10,000 official validation images. We use native 64×64 inputs and a ViT with 4×4 patches, width 192, 12 blocks, six heads, and MLP ratio 4. The official validation set is used for model and checkpoint selection, and no additional test split is used. The architectural conventions below also apply to ImageNet in Section L.5.

The backbone retains a class token and positional embeddings, but classifies by mean pooling over patch tokens followed by normalization and a linear head. It uses normalized query/key vectors, RMSNorm within the residual blocks, learned branch scaling, and a smooth gated MLP activation. The architecture omits both the distillation token and the convolutional token-mixing stage. These modifications apply equally to the local controls and GGM, ensuring that comparisons are based on the same modified ViT backbone rather than an unchanged DeiT implementation.

Conceptual Layer and Precision Scope. Each GGM map compares normalized input and weight directions through low-bit random measurements, followed by one learned output gain γ_o per channel. Direction normalization is intrinsic to the angular construction, and there are no separate learned input and weight scales. Training uses the proposed angular gradient. This experimental section refers to the layer construction developed in the method section rather than restating its implementation.

GGM replaces all 72 query/key/value/output and MLP projections. Patch embedding, class and positional tokens, normalization, nonlinearities, residual operations, and the classifier remain FP. Attention-score products, softmax, and attention–value aggregation also remain FP. Thus the low-bit labels describe the learned projections, not every operation in the network.

We use $(\nu_w/\nu_a) \times \nu$, with $\nu = N/p$, where p is the projection input dimension and N is the number of measurements. An L-level quantizer has cardinality precision $\log_2 L$. Here $L = 2$ and $\nu = 1$.

Conventional FP controls use ordinary linear body maps. LSQ-style layers quantize directly without a random-measurement dimension; their $\times 1$ denotes the unexpanded backbone.

Experimental Comparison and Evaluation. Low-bit models use pretrained initialization and auxiliary supervision. Within each GGM/LSQ-style comparison, the backbone, projection coverage, training budget, and supervision are matched. Initialization is specific to each operator family, so these comparisons evaluate the specified training procedures rather than isolating the quantizer alone. The conventional FP reference is separately tuned. No LSQ-specific recipe search or multi-seed replication is included.

Validation uses the original 64-pixel images with dataset normalization. We report single-seed top-1 accuracy over the full official validation set, using the same recorded evaluation convention for the local comparisons. Raw weights and their exponential moving average (EMA) are distinguished; the columns below are separately maximized periodic scores and need not come from the same checkpoint. Model selection on this validation set does not provide an independent test estimate.

Table 11: Selected TinyImageNet results (single seed). FP denotes the conventional linear body. Raw and EMA are best periodic top-1 accuracies (%) under the stated projection coverage.

Network/Method	Config.	Raw	EMA
ViT (FP), tuned	$32/32$	54.10	55.10
LSQ-style	$(1/1) \times 1$	40.52	40.81
GGM (Ours)	$(1/1) \times 1$	**51.25**	**51.70**

Results and Local Baseline. GGM reaches 51.70% EMA top-1, exceeding the specified LSQ-style control by 10.89 points and remaining 3.40 points below the conventional FP reference. The selected GGM checkpoint reproduces its periodic score under the matching evaluation runtime. These are selected single-seed results, not estimates of variation across repeated training runs.

Our local baseline uses learned-step quantization. Its binary version is a scaled-sign adaptation, labeled LSQ-style rather than an exact reproduction of published LSQ [5]. The same distinction applies to ImageNet. Completed training histories and checkpoint integrity have been verified for these controls, although independent reevaluations of the saved LSQ-style checkpoints are not documented in the available evidence. The reported advantage is therefore limited to these specific controls and does not establish superiority over a separately tuned LSQ pipeline.

Relation to Competing Methods. BiViT [7] reports TinyImageNet top-1 of 58.66% for Swin-T and 52.21% for NesT-T with binary attention and MLP weights/activations. Those results use different architectures, larger input resolution, and a different training protocol. Its higher 71.20%/69.83% variants retain FP MLP activations and must be distinguished from the fully binary MLP rows. SI-BiViT [24] studies spatial-interaction branches and decoupled training. These methods provide context for architectural adaptation and quantization coverage; they are not matched controls for our native-resolution ViT. Our controlled evidence is the local LSQ-style comparison, and we make no TinyImageNet state-of-the-art claim. Section L.5 extends this discussion to ImageNet.

L.5 ImageNet-1K Vision Transformer Experiments

Dataset, Backbone, and Measurement Scope. ImageNet-1K supplies 1,281,167 training images and 50,000 validation images over 1,000 classes. We retain the architecture, 72-map quantization coverage, FP-operation policy, and comparison conventions of Section L.4, changing resolution/patch size/width to $224/16/384$. Depth 12, six heads, and MLP ratio 4 are unchanged. These dimensions follow DeiT's ViT-S/16 scale [18]; the normalization, activation, and pooling choices described above make this a modified ViT backbone.

All GGM variants use $\nu = 1$. Binary, three-, four-, and five-level representations give $(1/1) \times 1$, $(\log_2 3/\log_2 3) \times 1$, $(2/2) \times 1$, and $(\log_2 5/\log_2 5) \times 1$, respectively. These are symmetric codebooks applied to the projected features. The cardinality precisions $\log_2 3 \simeq 1.585$ and $\log_2 5 \simeq 2.322$ require 2 and 3 bits under fixed-width storage. The notation does not by itself establish realized memory savings or hardware speedups.

Evaluation and Selected Results. The pretrained-initialization and auxiliary-supervision disclosure in Section L.4 applies here. Each local GGM/LSQ-style pair uses the same backbone, projection coverage, training budget, and supervision, but operator-specific initialization. The conventional FP model is a separately tuned reference. Validation resizes the shorter side to 256 pixels and center-crops to 224 before normalization. Table 12 reports best periodic EMA top-1 on all 50,000 validation images for completed single-seed runs. The validation set is also used for model selection and no independent test score or multi-seed significance claim is implied.

Table 12: Selected ImageNet-1K results. FP denotes conventional linear body maps. All rows report best periodic EMA top-1 (%) for completed runs, with the shared projection coverage described in Section L.4.

Network/Method	Config.	EMA top-1
ViT (FP), tuned	$32/32$	79.342
LSQ-style	$(1/1) \times 1$	46.470
LSQ-style	$(2/2) \times 1$	73.594
GGM (Ours)	$(1/1) \times 1$	**66.834**
GGM (Ours)	$(\log_2 3/\log_2 3) \times 1$	**73.464**
GGM (Ours)	$(2/2) \times 1$	**76.128**
GGM (Ours)	$(\log_2 5/\log_2 5) \times 1$	**77.168**

What the Local Comparisons Establish. At $(1/1) \times 1$, GGM reaches 66.834%, compared with 46.470% for LSQ-style, a 20.364-point difference. At $(2/2) \times 1$, the corresponding scores are 76.128% and 73.594%, a 2.534-point difference. Three and five levels reach 73.464% and 77.168%; the latter is 2.174 points below the conventional FP reference. These selected configurations demonstrate the achievable accuracy across precisions, but they do not constitute a precision-only ablation with every training choice fixed. Matched LSQ-style controls for three and five levels remain unavailable. The baseline adaptation, family-specific initialization, evaluation, and single-seed qualifications in Section L.4 apply throughout.

Published Competitors and Interpretation. Table 13 separates published results from our measurements. Q-ViT addresses quantized attention and uses 8-bit first/last layers [10]. Its DeiT-S results and LSQ reproduction are not measurements of our backbone. SI-BiViT adds spatial-interaction branches and decoupled training [24]. Its ViT-S and DeiT-S accuracies are external published references, not local reproductions. BiViT combines softmax-aware attention binarization with staged optimization, while its 75.6% Swin-S result retains FP MLP activations [7].

BHViT uses a convolution–Transformer hybrid designed for binarization, with convolutional token mixing in early stages, attention in later stages, and convolutional downsampling between stages [6]. Its higher binary accuracy is therefore reported for a different architecture and quantization scope. In contrast, GGM retains self-attention/MLP blocks throughout, without convolutional token mixing. This distinction clarifies the setting of our experiments without attributing BHViT's performance solely to architecture or predicting GGM's performance on its hybrid backbone.

Our binary result is numerically above the cited SI-BiViT ViT-S result and below BHViT-Small, while our two-bit result is numerically above the cited Q-ViT result. Architecture, initialization, supervision, training budget, and FP coverage differ across these papers, preventing a matched ranking of quantizers. The strongest conclusion is the measured improvement over the specified local LSQ-style controls within our shared ViT backbone, together with the selected multilevel accuracy results.

Table 13: Published ImageNet results for context, not a matched leaderboard. Precision is ν_w/ν_a; these methods have no GGM measurement ratio ν. BiViT's MLP activation precision is stated separately.

Method/source	Backbone / qualification	ν_w/ν_a	Top-1
SI-BiViT [24]	ViT-S	1/1	52.66
SI-BiViT [24]	DeiT-S	1/1	55.67
Q-ViT [10]	DeiT-S	2/2	72.1
LSQ in Q-ViT [10]	DeiT-S	2/2	68.0
BiViT [7]	Swin-S; MLP activations FP	1/1; 1/32	75.6
BHViT [6]	Small; convolution–Transformer	1/1	68.4
BHViT [6]	Same hybrid; downsampling FP	1/1	70.1

L.6 Transformer Text Classification

We evaluates GGM on Transformer models for text classification using the BiBERT [53] architecture on SST-2 and QNLI from the GLUE benchmark [57]. For both tasks, teacher and student models are initialized from pretrained DynaBERT [35] checkpoints following the BiBERT setup. For the student model, only the embedding checkpoint is reused, while the remaining parameters are initialized randomly and trained from scratch. We use 1-bit quantization for weights, embeddings, and input activations in the student model where embeddings are quantized according to BiBERT, while input activations and weights are quantized via GGM.

Among the GGM-based Transformer variants, the strongest performance is obtained when using BinaryNorm in place of LayerNorm and ReLU in place of GELU. BinaryNorm is a lightweight normalization layer that performs mean-centering followed by a learned affine transformation, without variance normalization. This avoids the square-root and division operations associated with LayerNorm while preserving activation centering. The distillation losses, including prediction distillation, intermediate representation distillation, and query/key/value distillation, are inherited from the BiBERT training pipeline and are not introduced specifically for GGM. We apply GGM to the linear layers with projection expansion ratios $\nu = 1.0, 2.0, 3.0$, while keeping the attention products $QK^\top$ and AV in full precision. Furthermore, maximum sequence length is 64 for SST-2 and 128 for QNLI.

For optimization, following BiBERT, we use BertAdam with linear warmup followed by linear decay, learning rate 1×10^{-4}, and warmup ratio 0.1. Weight decay is 0.01 for non-bias and non-normalization parameters and 0 for bias and normalization parameters. Both tasks use batch size 32 and are trained for 50 epochs. Evaluation is performed every 200 steps for SST-2 and every 1000 steps for QNLI, and the best performing checkpoint is selected based on validation performance. For this experiment we used a single H100 GPU and enabled mixed precision.

GGM substantially improves over prior binarized Transformer baselines on both tasks. On SST-2 at 1-bit and $\nu = 1.0$, GGM achieves 90.56%, improving over BiBERT by 1.86 points and at $\nu = 3.0$ leaves a 1.46-point gap to full precision. On QNLI, GGM reaches 86.24% at 1-bit and $\nu = 1.0$, improving over BiBERT by 13.64 points; see, Table 14 for detailed results.

Table 14: Classification accuracy on SST-2 and QNLI. GGM is built on top of the BiBERT training setup. Configurations are reported as $(\nu_w/\nu_a)\times$ projection expansion ratio, denoting weight, embedding, and activation bit-widths. * denotes XNOR-style fully binarized BERT.

Method	Config.	SST-2	QNLI
Full Precision	32/32	93.20	92.10
BinaryBERT	$(1/1) \times 1.0$	53.20	51.50
BERT-1bit*	$(1/1) \times 1.0$	77.60	66.40
BiBERT	$(1/1) \times 1.0$	88.70	72.60
GGM (Ours)	$(1/1) \times 1.0$	**90.56**	**86.24**
GGM (Ours)	$(1/1) \times 2.0$	**91.52**	**88.83**
GGM (Ours)	$(1/1) \times 3.0$	**91.74**	**89.73**

Table 15: Classification accuracy on PTB-XL and ADFTD. GGM uses the Medformer backbone with projection expansion ratio $\nu = 1.0$. Results are reported as mean $\pm$ standard deviation over multiple runs.

Method	Config.	PTB-XL	ADFTD
Crossformer	32/32	73.30 ± 0.14	50.45 ± 2.31
PatchTST	32/32	73.23 ± 0.25	44.37 ± 0.95
MTST	32/32	72.14 ± 0.27	45.60 ± 2.03
Medformer	32/32	72.87 ± 0.23	53.27 ± 1.54
DoReFa-Net	$(1/1) \times 1.0$	72.80 ± 0.06	52.41 ± 1.65
AdaBin	$(1/1) \times 1.0$	72.28 ± 0.17	49.26 ± 2.44
XNOR-Net	$(1/1) \times 1.0$	72.12 ± 0.08	52.56 ± 0.49
GGM (Ours)	$(1/1) \times 1.0$	$\mathbf{73.08 \pm 0.09}$	$\mathbf{53.39 \pm 0.29}$

L.7 Medical Time-Series Classification

We evaluate GGM on two large-scale medical time-series classification benchmarks: PTB-XL [56] for ECG classification and ADFTD [51] for EEG classification. We compare against strong full precision baselines as well as binary variants built on the same Medformer [60] backbone, including DoReFa-Net [68], AdaBin [19], and XNOR-Net [17].

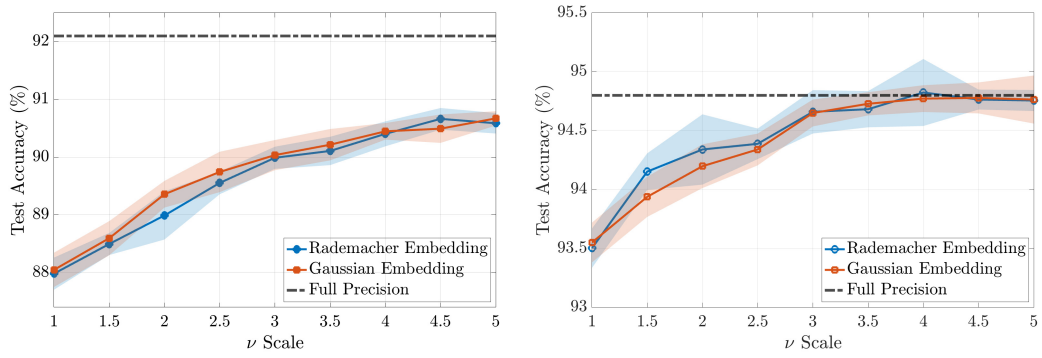
In the binary experiments, we preserve the Medformer architecture (with 6 encoder blocks, 1×1 feed-forward convolution, replacing the fully-connected layers, and classification pipeline) and replace the floating-point transformations inside the encoder with 1-bit binary counterparts. The embedding dimension $D = 128$, the hidden MLP dimension $d_{\text{ff}} = 256$ and the multi-scale patch configuration $\{2, 4, 8, 8, 16, 16, 16, 16, 32, \cdots, 32\}$ are kept the same as the baseline Medformer architecture. The PTB-XL experiments use an input sequence length of 250, while ADFTD uses the dataset-inferred sequence length. For the GGM variant, the query, key, value, and output projections in the attention blocks, as well as the two 1×1 feed-forward convolutions, are replaced by angular layers with $\nu = 1.0$; the encoder blocks use BinaryNorm in place of the standard LayerNorm, while the embedding module, final classification head, and training recipe follow the original Medformer design. In both tasks, models are trained with Adam optimizer, cross-entropy loss, stochastic weight averaging, batch size 20, early stopping with patience 10 with up to 20 and 100 epochs of training for PTB-XL and ADFTD, respectively. The learning rate was set to 1×10^{-3} for PTB-XL and 1×10^{-4} for ADFTD. For a fair comparison, we applied the same layer replacement procedure to the top 1-bit quantization baselines from the literature. Each experiment was reproduced on a single H100 GPU with 3 different seeds, and we report the mean accuracy and standard deviation across the seeds.

Table 15 gives the detailed results: GGM reaches 73.08 ± 0.09 on PTB-XL and 53.39 ± 0.29 on ADFTD obtaining the highest performance on both datasets among the low-precision methods while remaining competitive with the floating-point baselines. These results suggest that GGM can preserve or improve discriminative performance even under aggressive quantization.

L.8 Additional Analyses

L.8.1 Projection Distribution: Gaussian vs. Rademacher

In Figure 6, we showcase whether the distribution used to generate the projection matrix $\mathcal{G}$ affects GGM performance. We compare Gaussian and Rademacher matrices on CIFAR-10 as ν increases from 1.0 to 5.0. For this experiment we ran ResNet-18 and ResNet-20 CNN architectures on 10 different seeds for each projection expansion ratio ν. In both architectures the stem convolution and the classification head were kept full precision while all the hidden layers were 1-bit quantized following GGM, where projection matrices $\mathcal{G}$ in each angular layer are drawn either from standard Gaussian or Rademacher distribution. We trained each model for 300 epochs, base learning rate 3×10^{-3} with OneCycle learning rate schedule, weight decay 0.02, label smoothing 0.05, and batch size 256. Figure 6 shows the trend for both ResNet-18 and ResNet20 architectures. Rademacher and Gaussian choices yield similar trends in the two architectures, with accuracy improving as ν increases. The Rademacher distribution is particularly attractive as it is more storage-efficient, yet does not noticeably degrade accuracy in this setting.



(a) ResNet-20 on CIFAR-10.

(b) ResNet-18 on CIFAR-10.

Figure 6: Comparison of Gaussian and Rademacher choices for the projection matrix in the random sign embedding.

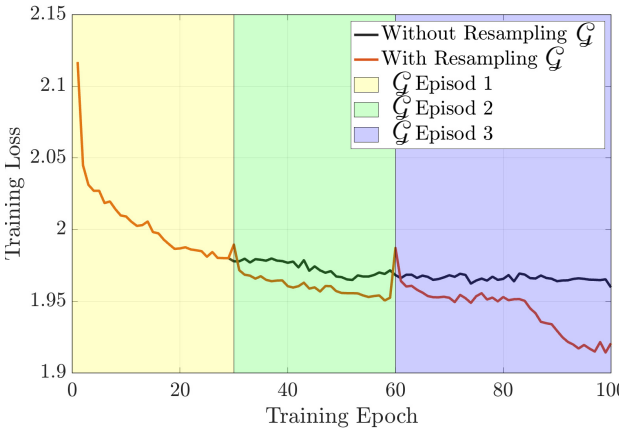


Figure 7: Training loss on CIFAR-10 for a small convolutional network with and without $\mathcal{G}$-matrix resampling. Resampling at epochs 30 and 60 produces brief loss spikes followed by improved convergence and lower final training loss.

L.8.2 Resampling the Projection Matrix

We study the effect of resampling the projection matrix $\mathcal{G}$ during training. We use a small convolutional network on CIFAR-10 with identical seeded initialization across runs and compare a fixed-$\mathcal{G}$ baseline against a setting where $\mathcal{G}$ is redrawn at epochs 30 and 60 (Figure 7).

Although the experiments reported in the main text do not rely on resampling, this diagnostic illustrates a useful property of the random embedding mechanism. When optimization becomes trapped in an unfavorable basin and the current stepsize is insufficient to move the iterate out, resampling $\mathcal{G}$ changes the binary random-feature map and can redirect the training trajectory. The resulting perturbation may temporarily increase the loss, but it can also move the model into a more favorable basin, after which training resumes and converges to a lower loss. Thus, the randomness in angular layers plays a dual role: it defines the angular random-feature geometry used for low-bit computation, and, when resampled, can also promote broader exploration of the optimization landscape.

L.8.3 Projection Expansion vs. STE Width Scaling

The projection expansion ratio ν controls the dimension of the GGM random sign embedding and therefore the amount of binary capacity introduced by the method, while leaving the number of full precision trainable weights unchanged. To compare this with conventional STE scaling, we increase capacity in STE models by widening the network before binarization. Figure 8 compares the two mechanisms on CIFAR-10.

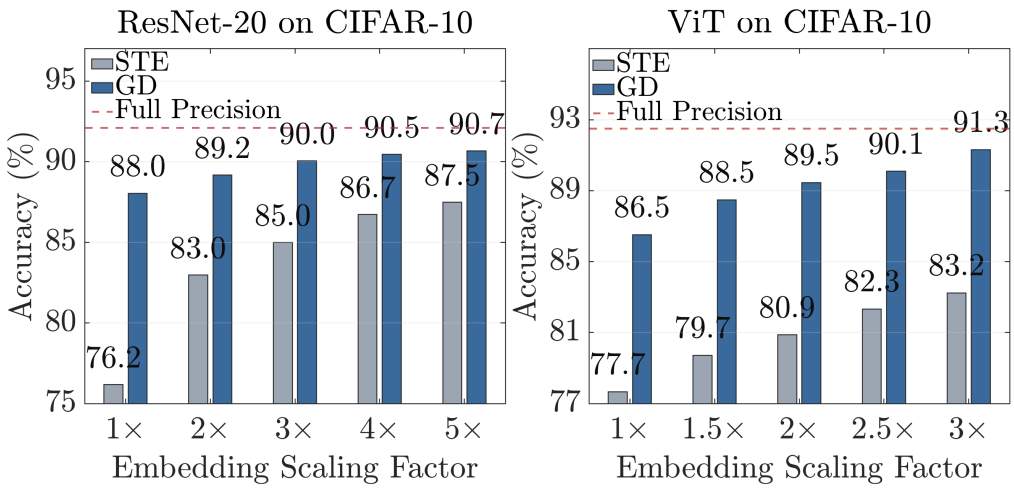


Figure 8: Comparison between STE width scaling and GGM projection expansion on CIFAR-10. For STE, ν denotes width expansion before binarization; for GGM, ν denotes projection expansion $N = \nu p$.

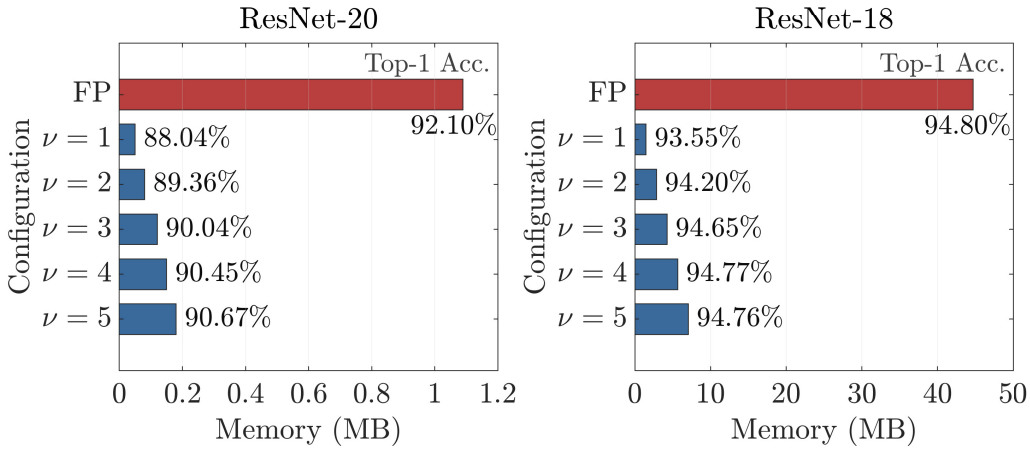


Figure 9: Memory–accuracy trade-off for GGM on CIFAR-10. Increasing ν improves accuracy while retaining a substantial compression advantage relative to full precision.

GGM uses additional binary capacity more effectively than width-scaled STE. On ResNet-20, GGM at $\nu = 1.0$ already exceeds the $5\times$ STE model, and increasing ν further narrows the gap to full precision. On ViT, GGM improves from 86.51% at $\nu = 1.0$ to 91.31% at $\nu = 3.0$, whereas width-scaled STE reaches only 83.24% at $3\times$ width.

L.8.4 Memory–Accuracy Trade-off

We also analyze the memory–accuracy trade-off induced by ν. Increasing ν improves predictive performance but increases the memory footprint of the low-bit representation. Figure 9 summarizes this trade-off for ResNet-20 and ResNet-18 on CIFAR-10.

For ResNet-20, the full precision model requires 1.09 MB and achieves 92.1% accuracy. GGM with $\nu = 1.0$ uses only 0.05 MB while achieving 88.51%, and increasing ν to 5.0 uses 0.18 MB while reaching 90.67%. For ResNet-18, the full precision model uses 44.71 MB and achieves 94.8%; GGM with $\nu = 1.0$ uses 1.48 MB while reaching 93.58%, and larger ν values approach full precision accuracy at a small fraction of the memory cost.

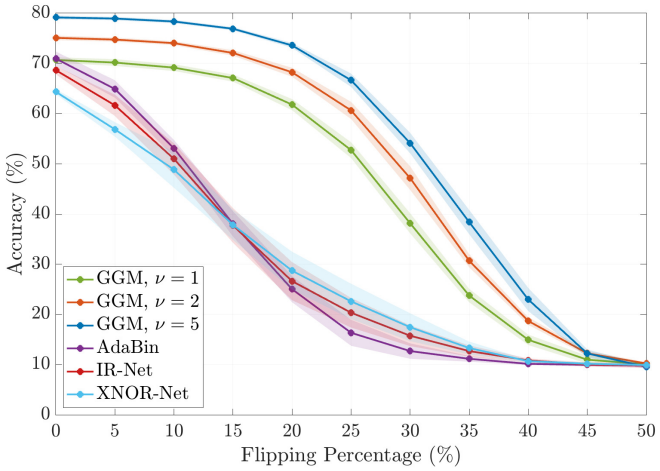


Figure 10: Robustness to binary weight perturbations. Accuracy is evaluated after independently flipping the sign of each learned binary weight with probability p, for perturbation levels from 0% to 50%, in a compact CIFAR-10 convolutional network.

L.8.5 Robustness to Binary Weight Perturbations

As discussed in the main text, the randomness in GGM can also improve the robustness of the trained models. To evaluate stability under discrete parameter corruption, we perturb trained binary weights after training and measure the resulting accuracy. For this experiment, we use a compact CIFAR-10 convolutional network with a full precision input convolution, followed by one binary convolution and two binary linear layers; the final classifier is kept full precision. Perturbations are applied only to the learned binary-weight layers. For each perturbation level $p \in [0, 0.5]$, we use a fixed random mask and independently flip the sign of each binary weight with probability p. Figure 10 reports the resulting robustness curves.

Finally, this random-feature averaging helps explain perturbation robustness. In STE baselines, a flipped bit directly corrupts a learned coordinatewise parameter; in GGM, a flipped bit corrupts only one term in an average over $N = \nu p$ projected coordinates. Under independent weight-bit flips with rate η, a binary product is attenuated in expectation by approximately $1 - 2\eta$, so perturbations mainly inject noise or shrink the kernel estimate rather than destroying a specific learned feature.

L.9 Gradient–Correlation Profiles for GGM Kernels

The theory in Section J gives the correlation law and derivative identities for ternary and more general quantized random embeddings. Here we use those identities to interpret the gradient profiles plotted in Figure 11. The key distinction is that the figure does not plot a scalar quantizer surrogate derivative. It plots the vector-gradient magnitude induced by the GGM kernel as a function of the normalized correlation $\rho(w, x) = \frac{w^\top x}{\|w\| \, \|x\|}$. For a B-bit kernel $\mathcal{K}_B(\rho)$,

$$\|\nabla_w \mathcal{K}_B(\rho(w, x))\| = |\mathcal{K}'_B(\rho)| \, \|\nabla_w \rho(w, x)\|,$$

and since

$$\nabla_w \rho(w, x) = \frac{1}{\|w\|} \left(\hat{x} - \rho \hat{w}\right), \qquad \|\nabla_w \rho(w, x)\| = \frac{\sqrt{1 - \rho^2}}{\|w\|},$$

the unit-norm gradient profile will follow

$$g_B(\rho) = |\mathcal{K}'_B(\rho)| \sqrt{1 - \rho^2}.$$

Thus, the plotted curves combine both the kernel derivative from the Gaussian correlation law and the geometry of normalized cosine similarity.

For the binary case, $\mathcal{K}_1(\rho) = \frac{2}{\pi} \arcsin(\rho)$, so

$$\mathcal{K}'_1(\rho) = \frac{2}{\pi \sqrt{1 - \rho^2}}, \qquad g_1(\rho) = \frac{2}{\pi}.$$

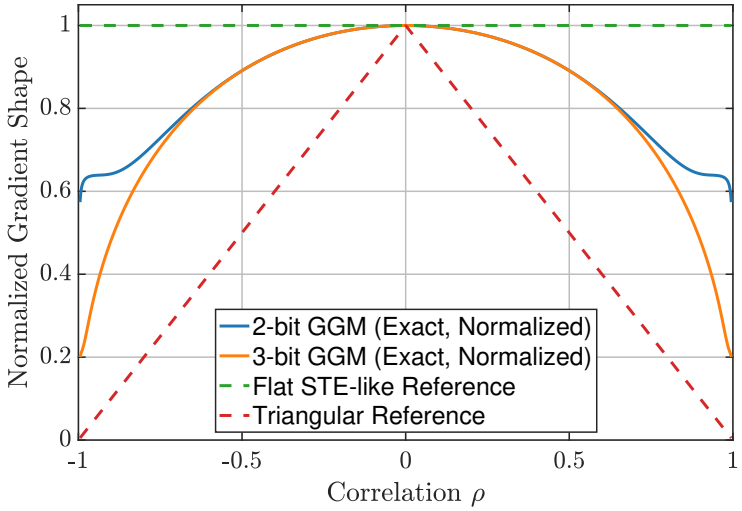


Figure 11: Normalized multibit G-gradient shapes as functions of the normalized correlation ρ. The plotted quantity is $g_B(\rho) = \mathcal{K}'_B(\rho)\sqrt{1-\rho^2}$, normalized to unit peak over $(-1, 1)$, where $\mathcal{K}'_B(\rho)$ is computed from the exact BVN-based kernel construction used in the implementation. The flat and triangular curves are reference shapes only and are not exact multibit STE gradients.

Hence the apparent singularity of $\mathcal{K}'_1(\rho)$ near $|\rho| = 1$ is exactly canceled by the cosine-geometry factor $\sqrt{1-\rho^2}$. For multibit kernels, $\mathcal{K}'_B(\rho)$ is computed using the bivariate-normal density expressions from Section J, with the same clipped symmetric quantization intervals and reconstruction levels as the forward pass. The kernel $\mathcal{K}_B$ is then obtained by numerically integrating $\mathcal{K}'_B$ on the ρ-grid with the convention $\mathcal{K}_B(0) = 0$.

In Figure 11, we have plotted the normalized profile

$$\tilde{g}_B(\rho) = \frac{g_B(\rho)}{\max_{\rho \in (-1,1)} g_B(\rho')},$$

for $B \in \{2, 3\}$. The curves are largest at intermediate correlations and decay smoothly near $|\rho| = 1$, indicating that GGM places its strongest local angular learning signal on directions that are not yet saturated. The flat and triangular curves are included only as reference shapes; they are not exact STE gradients.

L.10 End-to-End Efficiency of Packed Angular Vision Transformers

We complement the accuracy results of Angular Nets with end-to-end efficiency measurements for binary-weight/binary-activation Vision Transformers on CIFAR-10 and ImageNet-1K. We measure latency, throughput, GPU energy, persistent model state, and runtime composition, including projection overhead and retained floating-point operations. We also vary the projection expansion ratio $\nu = N/p$ from one to four in training and inference. The principal deployment setting is $(1/1) \times 1$ ($N = p$). Following the notation for Angular Layers, $\mathcal{G} \in \mathbb{R}^{N \times p}$ denotes a random projection matrix and $\mathcal{Q}_1(z) = \mathrm{sign}(z)$ the binary quantizer. We distinguish optimized deployment measurements from controlled expansion sweeps and from accuracy measurements obtained with different projection operators.

L.10.1 CIFAR-10

Architecture and Precision Allocation. The evaluated ViT processes 32×32 images into 64 patch tokens and one class token, with 12 width-192 Transformer blocks, six 32-dimensional attention heads, a $192 \to 768 \to 192$ OddGate MLP, and ten output classes. GGM replaces all six learned projections per block ($W_Q, W_K, W_V, W_O, W_{\mathrm{up}}, W_{\mathrm{down}}$), giving 72 logical body matrices implemented as 48 packed modules through Q/K/V grouping. For each body matrix W, inference

stores precomputed one-bit codes $\mathcal{Q}_1(\hat{W}\mathcal{G}^\top)$ of its normalized weight rows and one FP32 gain per output row. The activation codes $\mathcal{Q}_1(\mathcal{G}\hat{x})$ of the normalized input are generated and consumed on chip using XOR–popcount operations. The structured projection uses seeded sign diagonals and two block-32 Hadamard passes without forming a dense $\mathcal{G}$. The residual stream, Flash/SDPA attention, patch embedding, normalization, residual coefficients, and classifier retain BF16/FP32 arithmetic. Thus the network is binary in its attention/MLP *projections*, not in every operation.

End-to-End Performance. We benchmark on an NVIDIA H100 80 GB (PyTorch 2.12.0, CUDA 12.6) using preloaded inputs, 25 warm-ups, 800 CUDA-event measurements, CUDA Graph replay, and eight-second NVML energy measurements. The TorchInductor BF16 comparator has the same topology, fused QKV/SDPA, and final normalization/classifier path. At batch one, packed inference reduces median latency from 0.335776 to 0.228096 ms/image ($1.472\times$), increases throughput from $2{,}978.2$ to $4{,}384.1$ images/s, and reduces gross energy from 0.048241 to 0.033058 J/image (31.5%). Idle-adjusted energy decreases from 0.007801 to 0.005697 J/image (27.0%). Every module and all online projection/packing overhead are included (Fig. 12).

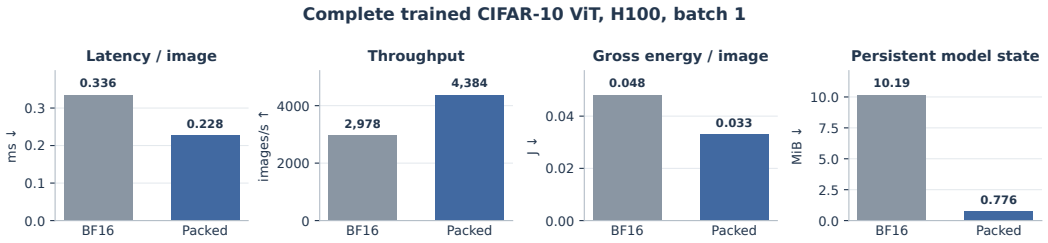


Figure 12: Complete CIFAR-10 ViT on H100, batch one: latency, throughput, gross GPU energy, and persistent model state.

Accuracy and Comparison Boundary. The independently evaluated packed model achieves 85.70% top-1 accuracy on all 10,000 test images. The corresponding training results report 85.51% for the hard structured student and 90.34% for its continuous-kernel teacher. The packed implementation agrees exactly with its reference logits. The timed BF16 baseline has matched architecture and kernel shapes but random body values, so the speedup is a *systems* comparison, not a comparison of two trained models at matched accuracy. This six-head deployment model is also distinct from the three-head controlled CIFAR-10 ViT used elsewhere in the paper.

Storage and Runtime Memory. The body contains $5{,}308{,}416$ weights, corresponding to 10.125 MiB in BF16 or 0.633 MiB as one-bit codes. With gains and retained FP components, the complete persistent state is 10.189 versus 0.776 MiB ($13.13\times$ compression). A transposed packed-code view and 1,920-byte projection cache increase the packed *resident* tensor state to 1.411 MiB, still $7.22\times$ smaller. The serialized inference representation occupies 0.832 MiB, compared with 20.584 MiB for the FP32 training checkpoint ($24.74\times$). This disk ratio uses an FP32 comparator and is distinct from the BF16 inference-state comparison. In isolated processes, graph-resident CUDA allocations are 106.221 versus 1.452 MiB and peak allocations 106.391 versus 1.596 MiB. Such compiler-dependent allocations must not be confused with persistent model storage. Training may still retain latent FP weights and optimizer state, so these numbers do not imply QAT-memory savings (Fig. 13).

Activation Movement and Execution Attribution. We examined activation movement to understand how much of the end-to-end cost remains after replacing the attention and MLP projections with packed operations. Although the projected codes are binary, we retain the intermediate Q/K/V representations and attention context in BF16. The residual stream also remains in BF16 and occupies $65 \times 192 \times 2 = 24{,}960$ bytes per image, which we reuse sequentially across blocks. To reduce the traffic between attention and the MLP, we fused the post-attention boundary with the following MLP, keeping the BF16-rounded attention output, the 768-wide hidden activation, and the packed projection signs on chip. This change reduced inference latency from 0.248928 to 0.228096 ms (8.4%) without changing the numerical output.

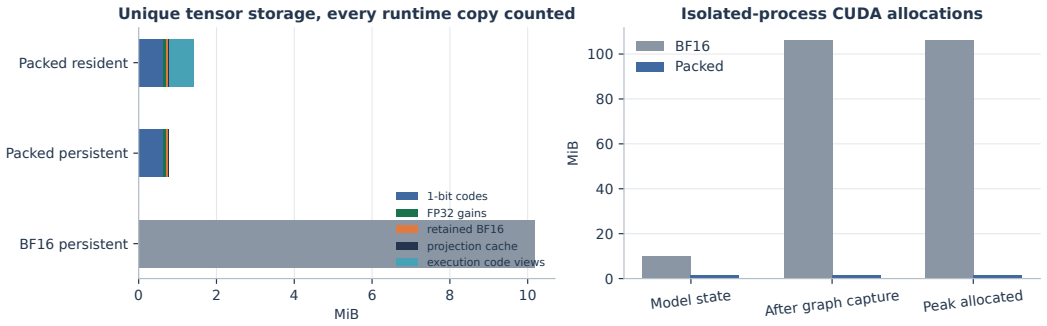


Figure 13: CIFAR-10: unique tensor storage, including execution copies (left), and isolated-process CUDA allocations (right).

To identify the remaining bottlenecks, we profiled 50 forward passes and measured 0.224993 ms per image. The packed QKV kernels account for 19.99% of CUDA time, while the fused attention-output/MLP kernels contribute another 41.18%. Flash attention accounts for 28.10%, with the final normalization and classifier (4.57%), token and layout operations (3.63%), and patch convolution (2.54%) making up the remainder. These timings already include the Hadamard projections, sign packing, row gains, activations, and residual epilogues. At the operation level, we count 345.047 million binary sign-pair products per image, evaluated as 10.783 million 32-bit popcount terms, together with 10.483 million projection additions and subtractions. The retained BF16 attention, patch embedding, and classifier require 19.469, 0.590, and 0.00192 million MACs, respectively (Fig. 14).

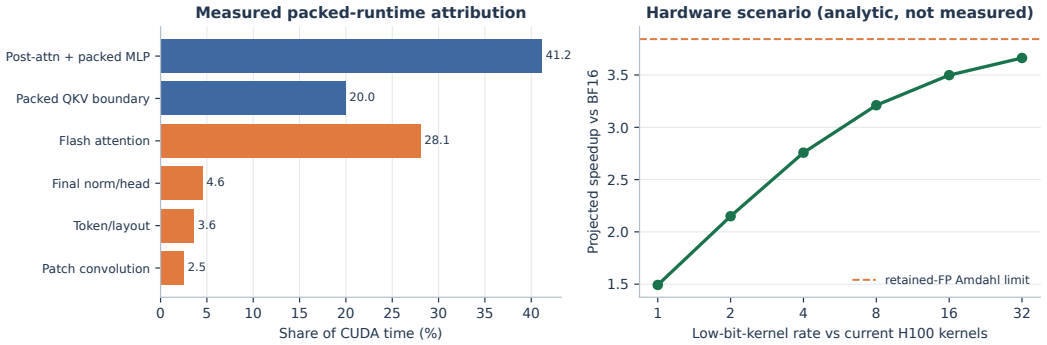


Figure 14: CIFAR-10 measured CUDA-time attribution (left) and analytic, not measured, low-bit-hardware scaling (right).

Batch-size Crossover. We varied the inference batch size to determine when the packed representation provides an end-to-end benefit. In the saved sweep, we obtained speedups of $1.464\times$, $1.478\times$, $1.513\times$, $1.515\times$, and $1.188\times$ for batches 1, 2, 4, 8, and 16, respectively. At batch 32, the ratio falls to $0.940\times$, making the BF16 implementation faster. We also found that gross device energy reaches parity earlier, between batches 8 and 16. This behavior is consistent with the increased reuse of dense weights and better Tensor Core utilization at larger batches. It suggests that the main benefit of our packed implementation lies in the small-batch, latency-sensitive regime rather than across all throughput settings (Fig. 15).

Projection Expansion and its Measured Cost. We next investigated how increasing the projection expansion ratio affects the cost of a complete model. For this controlled study, we varied $\nu \in \{1, 2, 3, 4\}$ and implemented the random embedding through seeded, two-pass Hadamard transforms

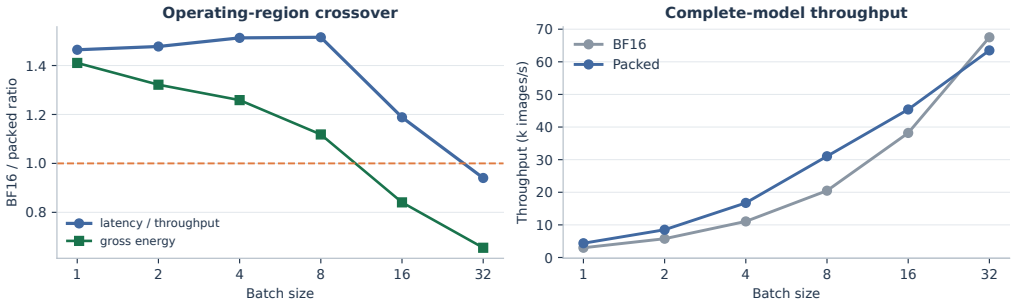


Figure 15: CIFAR-10 batch sweep: latency/energy ratios and full-model throughput.

on blocks of width $b = 32$. With $H_b H_b^\top = bI$, each frame is defined by

$$\mathcal{G}_f x = \mathrm{vec}\left(\frac{1}{\sqrt{b}} H_b D_{2,f} H_b D_{1,f} x^{(j)} \right)_{j=1}^{p/b}, \qquad \mathcal{G} = \begin{bmatrix} \mathcal{G}_1 \\ \vdots \\ \mathcal{G}_\nu \end{bmatrix}, \quad N = \nu p. \tag{40}$$

We generate the sign diagonals from shared seeds instead of storing a dense $\mathcal{G}$, and we precompute the packed weight codes before inference. For an $m \times p$ layer, this construction requires $O(2\nu p \log_2 b)$ projection additions and subtractions per token and νmp binary sign-pair products, with νmp bits of weight-code storage. Hence the projected work grows linearly with ν, but its contribution to the total runtime must be measured alongside the operations that remain in floating point.

We therefore measured complete-model training and inference, as shown in Table 16. At batch 64, increasing ν from one to four raised the QAT step time by 8.2%. Relative to the compiled BF16 training step of 8.063 ms, the resulting QAT steps were 1.70–$1.84\times$ slower. In the corresponding batch-one inference sweep, latency rose by 24.9% and gross energy per image by 34.4%, while persistent model state grew by $3.44\times$. We used the same portable implementation for every expansion factor to isolate this effect. Its $\nu = 1$ latency is higher than the 0.228096 ms optimized result because it does not use the same degree of fusion, so only within-sweep ratios are comparable. For each ν, we also verified exact agreement with the corresponding hard-forward operator. This numerical check does not establish accuracy for a newly trained higher-ν model.

Table 16: CIFAR-10 expansion sweep. Training includes forward/backward/SGD at batch 64. Portable packed inference uses batch one and a different implementation from the optimized headline.

	QAT, batch 64			Packed inference, batch 1			
ν	ms/step	images/s	peak MiB	ms/image	images/s	J/image	state MiB
1	13.693	4,674	488.4	0.670464	1,491.5	0.094756	0.777
2	14.593	4,386	489.5	0.717984	1,392.8	0.104415	1.410
3	14.644	4,370	488.1	0.789792	1,266.2	0.117368	2.043
4	14.813	4,321	487.4	0.837120	1,194.6	0.127364	2.675

Hardware Interpretation. We also examined what limits the speedup on the H100 and how the result might change on hardware with stronger native low-bit support. For this small ViT, the 50 MB L2 cache is large enough to hold the BF16 model state, and highly optimized BF16 Tensor Core kernels provide a strong comparator. At the measured token and channel dimensions, we found fused integer popcount kernels faster than the available native binary tiles. Of the 0.224993 ms profiled runtime, 0.137625 ms belongs to projected low-bit kernels and 0.087368 ms to retained floating-point and other fixed work. Using this measured breakdown, we estimate that a $4\times$, $8\times$, or $16\times$ acceleration of the low-bit portion alone would give end-to-end speedups of $2.76\times$, $3.21\times$, and $3.50\times$ over BF16, respectively. The unchanged work sets an Amdahl limit of $3.84\times$. These are analytic scenarios rather than hardware measurements. We expect a model whose BF16 weights exceed cache capacity to benefit more from reduced weight traffic at small batch sizes, although this expectation requires a separate model-scale evaluation.

L.10.2 ImageNet-1K

Architecture and Precision Allocation. We extended the packed implementation to an ImageNet-1K ViT to examine its behavior with longer token sequences and wider layers. The model processes 224×224 images into 197 tokens using 16×16 patches and has 12 width-384 Transformer blocks, six 64-dimensional attention heads, a $384 \to 1536 \to 384$ MLP, and a 1,000-class classifier. As in the CIFAR-10 model, we replace all 72 attention and MLP projection matrices with binary angular correlations. We precompute the one-bit weight codes $\mathcal{Q}_1(\hat{W}\mathcal{G}^\top)$ and generate the activation codes $\mathcal{Q}_1(\mathcal{G}\hat{x})$ on chip before applying packed popcount operations and FP32 row gains. We use $\nu = 1$, so the projection does not expand the binary code length. For the optimized runtime, we implement $\mathcal{G}$ using seeded block-32 Hadamard transforms with 3,840 bytes of shared sign diagonals rather than materializing a dense matrix. We leave the residual stream, Flash/SDPA attention, patch embedding, normalization, nonlinear and residual coefficients, and classifier in FP16 or FP32, as appropriate.

End-to-End Performance. We measured the complete ImageNet model on an H100 using preloaded inputs and CUDA Graph replay. After 20 warm-up iterations, we collected 200 paired CUDA-event measurements and evaluated device energy in separate eight-second NVML loops. For comparison, we compiled a BF16 model with the same topology, fused QKV projections, and Flash/SDPA attention using TorchInductor. At batch two, we measured a decrease in latency from 0.677536 to 0.393024 ms per batch, corresponding to a $1.724\times$ speedup. Throughput increased from $2{,}951.9$ to $5{,}088.7$ images/s. We also observed a 30.6% reduction in gross GPU energy, from 0.075477 to 0.052357 J/image, while idle-adjusted energy fell from 0.034337 to 0.028547 J/image (16.9%). These measurements cover the full forward pass, including the structured projections and all retained floating-point operations (Fig. 16).

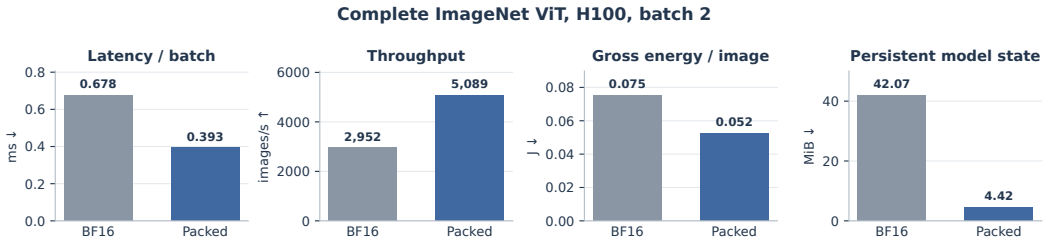


Figure 16: Complete ImageNet ViT on H100, batch two: latency, throughput, gross GPU energy, and persistent model state.

Accuracy and Projection Boundary. We evaluated numerical fidelity separately from the optimized ImageNet timing experiment because the available trained model and the fastest deployment path use different projection constructions. The trained model uses a dense Haar-orthogonal $\mathcal{G}$ and achieves 56.188% top-1 and 79.612% top-5 accuracy. When we replace its unpacked operator with the corresponding packed implementation, we obtain identical logits. For the faster 0.393024 ms runtime, however, we use a structured-Hadamard projection, and we have not measured the accuracy of a model trained with that exact operator. Similarly, the 66.834% binary ImageNet accuracy reported in our separate study does not correspond to this optimized structured deployment path. The BF16 timing comparator also uses random body values while preserving the architecture and execution shapes. We therefore interpret the ImageNet measurements as evidence of the efficiency of the deployment design rather than a joint accuracy-and-runtime comparison for one trained checkpoint.

Model Storage and CUDA Memory. To assess whether the reduction in binary weight storage remains significant after accounting for the floating-point modules, we measured both persistent model state and actual CUDA allocations. The 72 body matrices contain $21{,}233{,}664$ weights, requiring 40.500 MiB in BF16 but only 2.531 MiB as packed one-bit codes. Once we include the retained modules, row gains, and projection state, the complete persistent footprint falls from 42.067 to 4.419 MiB, a $9.52\times$ reduction. The serialized packed representation uses 4.268 MiB and does not include latent full-precision body weights or optimizer state.

We then measured allocations in isolated CUDA processes rather than inferring runtime memory directly from parameter sizes. With model and input tensors loaded, BF16 and packed inference

use 43.547 and 5.029 MiB, respectively. After graph capture, these values rise to 139.548 and 101.030 MiB, with peak allocations of 142.191 and 101.036 MiB. This shows that compiler and graph workspaces reduce the observed process-level saving to approximately 1.4×, despite the much larger reduction in persistent model state. We exclude the shared CUDA context from both measurements. Since quantization-aware training can still retain latent weights and optimizer state, we do not infer any training-memory saving from these inference results (Fig. 17).

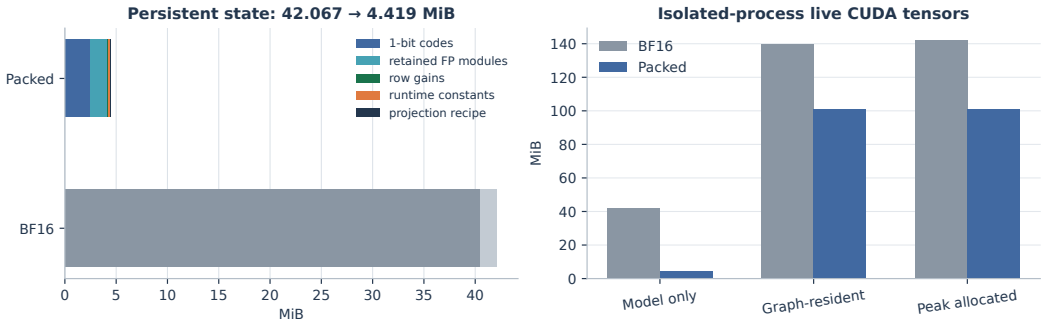


Figure 17: ImageNet persistent-state composition (left) versus actual CUDA allocations including graph workspaces (right).

Activation Traffic and CUDA-time Attribution. We examined the activation pathway to determine how much traffic remains after packing the angular projections. The FP16 residual stream occupies $197 \times 384 \times 2 = 151{,}296$ bytes per image and is reused across Transformer blocks. To avoid moving intermediate activations unnecessarily, we fused the post-attention projection with both MLP linear operations. The fused kernel keeps the 384-wide attention output and the 1,536-wide hidden activation on chip, avoiding approximately 0.721 MiB of global-memory writes per image per block and the corresponding reads.

We profiled 50 iterations of this implementation and measured 0.390696 ms per batch. The projected QKV operations account for 21.4% of CUDA time, while the fused output, MLP, and residual operations contribute 36.2%. These two components include the costs of centering, Hadamard projections, sign packing, row gains, and residual epilogues. We attribute 27.63% to retained Flash attention, 12.33% to patch convolution, and 2.47% to the remaining floating-point and layout operations. At the same time, the packed body evaluates 4.183 billion binary sign-pair products per image, implemented through 130.72 million 32-bit popcount terms, along with 31.77 million structured-projection operations. The retained FP16 attention, patch embedding, and classifier account for 29.81, 57.80, and 0.384 million MACs, respectively (Fig. 18).

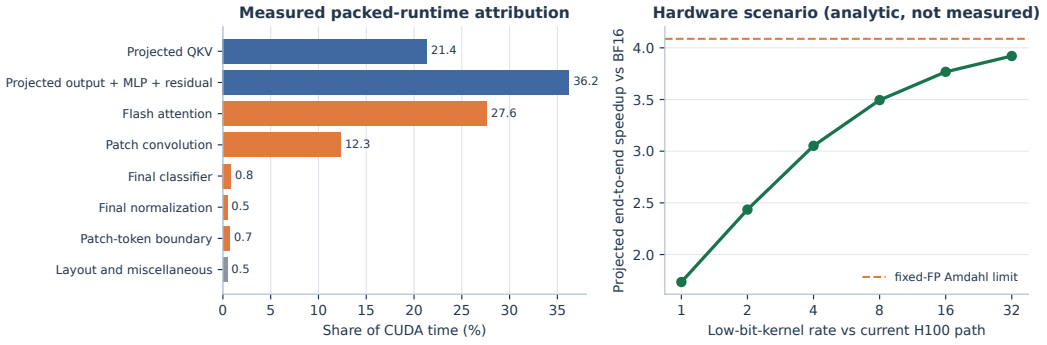


Figure 18: ImageNet measured runtime breakdown (left) and analytic low-bit-hardware scenario (right).

Batch-size Crossover. We repeated the batch-size sweep on ImageNet to determine whether the small-batch advantage persists for a wider ViT. We measured speedups of 1.542×, 1.724×, 1.513×,

$1.202\times$, and $1.182\times$ at batches 1, 2, 4, 6, and 8, respectively. The BF16 implementation becomes faster at batches 16 and 32, where the packed-to-BF16 speedup falls to $0.894\times$ and $0.776\times$. We attribute this crossover to improved dense-weight reuse and BF16 Tensor Core utilization as batch size grows. The H100's 50 MB L2 cache can also hold much of the 42.067 MiB BF16 model state. Our measurements therefore identify a favorable small-batch operating region, not a speedup that is independent of batch size (Fig. 19).

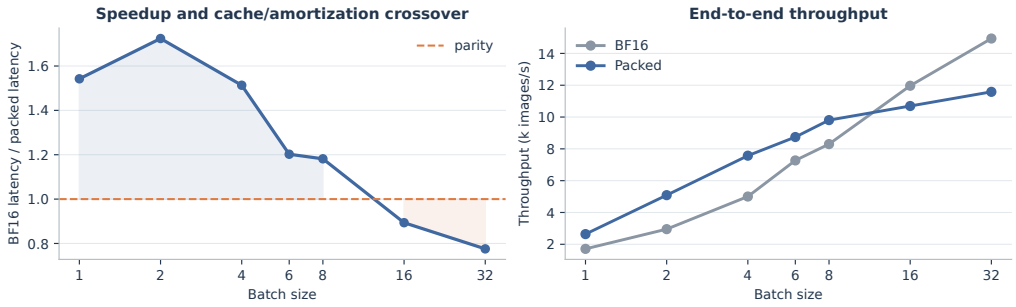


Figure 19: ImageNet batch sweep showing relative latency and complete-network throughput. The crossover occurs between batches 8 and 16.

Projection Expansion in Training and Inference. We repeated the projection-expansion experiment on ImageNet to determine whether the cost of additional random features becomes more significant at this scale. For $\nu \in \{1, 2, 3, 4\}$, we used the two-pass block-32 construction in (40) for input widths $p = 384$ and 1536. This controlled implementation differs from the optimized $\nu = 1$ runtime, which uses a more tightly fused, one-pass Hadamard projection. In both cases, the packed weight-code storage grows as νmp bits and the online activation projection grows linearly with ν.

We measured the complete QAT training step at batch two and found that increasing ν from one to four raises step time from 16.510 to 17.932 ms (8.6%), while peak allocation stays near 208 MiB. These QAT steps are 2.19–$2.38\times$ slower than the compiled BF16 training step of 7.535 ms. In the corresponding portable inference sweep, increasing ν raises latency by 72.4%, gross energy per image by 109.7%, and persistent state by $2.78\times$ (Table 17). We report these increases relative to the portable $\nu = 1$ time of 1.089584 ms, not the 0.393024 ms optimized result, because the two implementations have different fusion strategies. We verified exact agreement with each configuration's hard-forward logits, but did not use this systems sweep to estimate trained-model accuracy at $\nu > 1$.

Table 17: ImageNet expansion sweep. Both QAT (forward/backward/SGD) and portable packed inference use batch two. The inference rows do not use the optimized one-pass path.

	QAT, batch 2			Packed inference, batch 2			
ν	ms/step	images/s	peak MiB	ms/batch	images/s	J/image	state MiB
1	16.510	121.14	208.0	1.089584	1,835.6	0.125620	4.258
2	16.693	119.81	208.8	1.318784	1,516.5	0.169080	6.789
3	17.498	114.30	208.8	1.575168	1,269.7	0.214700	9.320
4	17.932	111.53	208.2	1.878624	1,064.6	0.263485	11.851

Hardware Outlook and Precision Trade-off. Finally, we used the ImageNet CUDA trace to assess which parts of the model could benefit from hardware specialized for packed binary computation. We measured 0.2249 ms in low-bit kernels and 0.1658 ms in retained floating-point or otherwise fixed work. If only the former were accelerated by $8\times$ or $16\times$, the resulting analytic end-to-end speedups over BF16 would be $3.49\times$ and $3.77\times$. The unchanged portion sets a $4.09\times$ Amdahl limit. We do not present these estimates as native-hardware or energy measurements. A larger model whose dense weights exceed the H100 cache may shift the batch-size crossover in favor of packed inference, but establishing that behavior will require additional experiments.

These results also inform how we choose the projection dimension and quantizer resolution. Increasing precision at fixed $N = p$ can preserve additional information without repeating the structured

projection, although it does not come at zero cost. In a bit-serial implementation, k-bit weights and ℓ-bit activations can require as many as $k\ell$ binary correlation planes and k stored weight planes. For example, $(2/2) \times 1$ and $(1/1) \times 4$ each involve four binary pair-product planes, but only the latter repeats the activation projection. We therefore view both bit-width and projection expansion as accuracy–efficiency choices that should be compared using end-to-end measurements.

M Limitations and Broader Impacts

While GGM demonstrates strong empirical performance, it introduces several trade-offs. First, the method can use random projection expansion, ν, to increase effective binary capacity, which increases intermediate compute and memory proportional to $N = \nu p$, even though the resulting operations remain low-bit and hardware-friendly (e.g., popcount-based). Notably, however, many of our experiments already show competitive or state-of-the-art performance with $\nu = 1$, i.e., without projection expansion, suggesting that expansion is a useful accuracy–efficiency knob rather than a strict requirement. Second, for multibit settings, the approach depends on precomputed kernel tables, which introduces additional implementation complexity and minor overhead compared to simple STE-based quantizers. Third, the current formulation is tailored to dot-product layers, and extending the same geometric construction to other operations, such as attention softmax or normalization layers, remains an interesting direction for future work.

This work aims to improve the efficiency of deep neural networks through geometry-aware low-bit representations. More efficient models can reduce memory usage, communication cost, and energy consumption during training and inference, potentially improving the accessibility and environmental sustainability of large-scale machine learning systems. In particular, low-bit models may enable deployment on resource-constrained hardware such as mobile, embedded, or edge devices. At the same time, increased efficiency can also lower the computational barrier for deploying large models at scale, including in applications that may raise privacy, surveillance, or misinformation concerns. The proposed methods are general-purpose and do not target any specific application domain; their societal impact therefore depends primarily on the downstream systems in which they are deployed.

Appendix References

[25] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. Binarybert: Pushing the limit of bert quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, 2021.

[26] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 02 2013.

[27] Wei-Kuo Chen, Nikos Dafnis, and Grigoris Paouris. Improved hölder and reverse hölder inequalities for gaussian random vectors. *Advances in Mathematics*, 280:643–689, 2015.

[28] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022.

[29] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020.

[30] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019.

[31] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[32] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4852–4861, 2019.

[33] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[34] Yefei He, Zhenyu Lou, Luoming Zhang, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Bivit: Extremely compressed binary vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5651–5663, 2023.

[35] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020.

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

[37] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4350–4359, 2019.

[38] Michael R Kosorok. *Introduction to empirical processes and semiparametric inference*. Springer, 2008.

[39] Phuoc-Hoan Charles Le and Xinlin Li. Binaryvit: Pushing binary vision transformers towards convolutional models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4665–4674, 2023.

[40] Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks, 2022.

[41] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. {BRECQ}: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021.

[42] Elliott H Lieb. Gaussian kernels have only gaussian maximizers. *Inventiones mathematicae*, 102(1):179–208, 1990.

[43] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

[44] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. *Advances in neural information processing systems*, 33:7474–7485, 2020.

[45] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *Advances in neural information processing systems*, 30, 2017.

[46] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. FQ-ViT: Post-training quantization for fully quantized vision transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 1173–1179, 2022.

[47] 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*, pages 143–159. Springer, 2020.

[48] 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 *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018.

[49] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits, 2024.

[50] Colin McDiarmid et al. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[51] Andreas Miltiadous, Katerina D. Tzimourta, Theodora Afrantou, Panagiotis Ioannidis, Nikolaos Grigoriadis, Dimitrios G. Tsalikakis, Pantelis Angelidis, Markos G. Tsipouras, Euripidis Glavas, Nikolaos Giannakeas, and Alexandros T. Tzallas. A dataset of scalp eeg recordings of alzheimer's disease, frontotemporal dementia and healthy subjects from routine eeg. *Data*, 8(6), 2023.

[52] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pages 7197–7206. PMLR, 2020.

[53] Haotong Qin, Yifu Ding, Mingyuan Zhang, Qinghua YAN, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. Bibert: Accurate fully binarized bert. In *International Conference on Learning Representations*, 2022.

[54] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2250–2259, 2020.

[55] Bodhisattva Sen. A gentle introduction to empirical process theory and applications. https://sites.stat.columbia.edu/bodhi/Talks/Emp-Proc-Lecture-Notes.pdf, 2022.

[56] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):154, 2020.

[57] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355, 2018.

[58] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models, 2023.

[59] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[60] Yihe Wang, Nan Huang, Taida Li, Yujun Yan, and Xiang Zhang. Medformer: A multi-granularity patching transformer for medical time-series classification. *Advances in Neural Information Processing Systems*, 37:36314–36341, 2024.

[61] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

[62] Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. Recu: Reviving the dead weights in binary neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5198–5208, 2021.

[63] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021.

[64] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183, 2022.

[65] Zhengqing Yuan, Rong Zhou, Hongyi Wang, Lifang He, Yanfang Ye, and Lichao Sun. Vit-1.58b: Mobile vision transformers in the 1-bit era, 2024.

[66] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018.

[67] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless CNNs with low-precision weights. In *International Conference on Learning Representations*, 2017.

[68] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2018.

[69] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. In *International Conference on Learning Representations*, 2017.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Theoretical contributions are presented in Section 2, with detailed proofs in the Appendix, and experimental results are presented in Section 4, with details and extended studies presented in the Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are explicitly discussed in Section M of the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Each result is precisely stated with all assumptions, and detailed proofs of each theorem. Supporting lemmas and references are clearly provided in the proofs presented in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Experimental settings and evaluation protocols are detailed in the paper and Appendix. Evaluation checkpoints and model weights for all reported models are publicly released, allowing independent verification of the results. The training code is not released due to intellectual-property considerations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: All benchmark datasets are publicly available, and evaluation checkpoints, model weights, and inference/evaluation code are released for the reported models. However, the training code is not publicly released due to intellectual-property considerations.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: To the extent needed, such details are provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Where applicable, we report error bars or variability information for repeated experiments. For comparisons with prior work, we follow the reporting convention of the corresponding source papers, which often report best or top accuracy under a fixed training protocol. Experimental settings and evaluation details are provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information on the computational resources used for the experiments is provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Broader Impacts are explicitly discussed in Section M of the Appendix.

Justification:

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work consists of fundamental research into low-bit neural networks and poses no such misuse risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing datasets, pretrained checkpoints, and third-party implementations are properly credited, with their applicable licenses and terms of use documented in the accompanying asset documentation. Our released AngularNet code is separately governed by the "AngularNet License Notice for Free Academic Research Use," Copyright 2026 Oregon State University.

    Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released model weights, evaluation checkpoints, and inference/evaluation code are accompanied by documentation and instructions for reproducing the reported evaluations. These assets are released under the AngularNet License Notice for Free Academic Research Use, with commercial use requiring prior authorization from Oregon State University.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Neither crowdsourcing nor human subjects is involved in the research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Neither crowdsourcing nor human subjects is involved in the research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The research does not involve use of LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.