PREFIX-POINT: AN ADAPTIVE NON-UNIFORM FORMAT FOR DIFFUSION MODEL QUANTIZATION

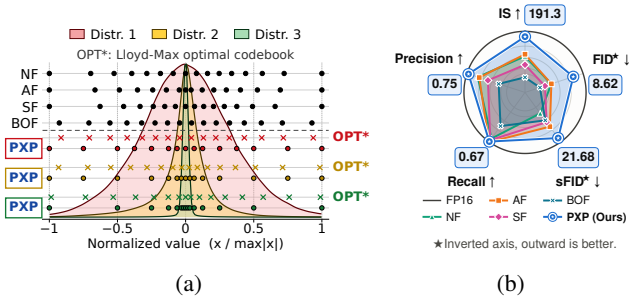
Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training quantization of diffusion models is challenging because activation distributions vary substantially across layers and denoising steps. Prior work mitigates this with step-aware scaling for INT quantization, but scaling adjusts only the range of a distribution and not its shape, so it cannot follow activations that shift between heavy-tailed shapes, with most values near zero, and light-tailed shapes spread evenly across the range. Non-uniform formats represent such shapes better, yet existing ones fix a single codebook at design time from an assumed distribution and reuse it throughout the network and therefore degrade generation quality sharply at low precision ($\leq$ 6 bits). We propose Prefix-Point (PXP), a format that divides the value range into exponent levels and sets the resolution of each level independently, allocating bit-width where it reduces quantization error most, and fitting this allocation separately to the distribution of each layer and denoising step. Since the resulting design space grows exponentially with bit-width, we cast level allocation as a multiple-choice knapsack problem and solve it exactly by dynamic programming. Its floating-point-like structure keeps decoding and the multiplier cheaper than in existing non-uniform formats. Across class-conditional, text-to-image and video diffusion, PXP lowers FID by up to $4.5\times$ over the strongest existing non-uniform format while being up to $6\times$ faster.

1 INTRODUCTION

Diffusion models (Peebles & Xie, 2023; Chen et al., 2024; Zheng et al., 2024) dominate high-quality image and video generation, but iterative denoising makes inference costly. Post-training quantization (PTQ) (Li et al., 2021; Xiao et al., 2023) offers a practical remedy, with integer (INT) formats as the de facto standard for their hardware efficiency. Diffusion models are especially hard to quantize, as activation distributions shift in both scale and shape across layers and denoising steps. Prior work (Chen et al., 2025; Wu et al., 2024; Shang et al., 2023; Xu et al., 2026) addresses this with step-aware parameters and increasingly sophisticated calibration and transformation, yet always within the INT format, leaving the numeric format itself largely unexplored.



(a) (b)

Figure 1: (a) 4-bit quantization levels for existing non-uniform formats compared to PXP (ours) which can adapt to the input distribution. **OPT*** represents MSE-optimal codebook of Lloyd-Max quantizer. (b) Comparison of W4A6 quantization using existing non-uniform formats vs. PXP (ours) on DiT-XL/2 (CFG=1.5) across 100 denoising steps.

Non-uniform formats (Dettmers et al., 2023; Blumenberg et al., 2026) are a natural alternative, offering two advantages over INT. At equal bit-width they cover a wider dynamic range while concentrating levels in high-density regions, directly reducing MSE, and unlike INT's fixed spacing, their level placement can in principle adapt to the target distribution. In practice, however, existing formats optimize placement for a fixed prior, typically a standard Gaussian, and rescaling can shift

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

these levels but not redistribute them. Since activations range from heavy-tailed, with most values near zero, to light-tailed and nearly uniform, a fixed level pattern matches only part of the network, and we show empirically that this largely negates the advantage of non-uniform formats.

We propose Prefix-Point (PXP), a format whose quantization levels are fitted to the data rather than to an assumed distribution. A fully adaptive Lloyd-Max codebook (Lloyd, 1982) minimizes mean squared error (MSE) but forfeits the efficiency that motivates quantization, requiring look-up-table (LUT) decoding and full-precision arithmetic. PXP retains this adaptivity within a hardware-friendly structure. It keeps the exponent-mantissa layout of floating point (Markstein, 2008) but encodes exponents as variable-length prefix codes, assigning shorter prefixes, and hence more mantissa bits, to the levels with the largest error. Because only exponents are table-encoded and decoded values remain narrow floats, both decoding and arithmetic stay cheap.

This adaptability comes at the cost of a search space that grows exponentially with bit-width, making exhaustive search infeasible beyond 6 bits. The problem, however, has exploitable structure. The total error decomposes into per-exponent level errors, each depending only on the values in that level and its mantissa width, so choosing one prefix length per level under a bit budget is exactly a multiple-choice knapsack problem, solved optimally by dynamic programming in under a second per tensor, over $10^5\times$ faster than exhaustive search, and entirely during calibration. Fig. 1a shows that existing formats keep the same level placement across three very different distributions, whereas PXP reshapes itself toward the Lloyd-Max optimum for each. On DiT-XL/2 over ImageNet (Fig. 1b), PXP outperforms every existing format and approaches FP16, lowering FID by up to $4.5\times$ over the strongest non-uniform baseline while being up to $6\times$ faster at W4A6, and this advantage carries over to text-to-image and video generation. We summarize our contributions below.

- **PXP: a fully adaptive quantization format.** We propose Prefix-Point, which dynamically allocates exponent levels and variable mantissa widths to minimize MSE for target distribution, without assuming a fixed prior.
- **Exact and efficient optimization.** We show that the objective separates across exponent levels, cast the allocation as a multiple-choice knapsack problem, and solve it optimally by dynamic programming in under a second per tensor, avoiding exhaustive search.
- **Extensive evaluation across tasks and architectures.** We evaluate PXP on class-conditional, text-to-image and video generation, spanning DiT, PixArt-Σ and OPEN-SORA, where it lowers FID by up to $4.5\times$ over the strongest existing non-uniform format.
- **Hardware efficiency.** RTL implementations of every format show that PXP's floating-point structure keeps decoding and arithmetic cheap, making it up to $6\times$ faster than existing non-uniform formats at W4A6 while significantly improving generation quality.

2 RELATED WORKS

Unlike quantization-aware training (Esser et al., 2020; Li et al., 2026; Zhang et al., 2026b; Dremov et al., 2026), post-training quantization (Li et al., 2021; Xiao et al., 2023) cannot retrain weights to absorb the induced error, which makes it far more sensitive to the choice of format. This motivates our work on an adaptive quantization format specifically designed for PTQ.

Diffusion Model Quantization: PTQ for diffusion models and DiTs faces challenges less pronounced in conventional vision models, as activation distributions shift across denoising steps and channel-wise variance imbalance hampers low-bit quantization (Wu et al., 2024; Shang et al., 2023). Existing methods address this with variance-equalized transformations and timestep-adaptive calibration (Shang et al., 2023; Wu et al., 2024; Xu et al., 2026; Chen et al., 2025), or exploit sample-level, frequency or semantic structure (Pham et al., 2026; Zeng et al., 2026; Zhang et al., 2026a). All of these refine calibration while keeping an integer representation. We instead target the format itself, making our approach orthogonal to and combinable with them.

Non-Uniform Quantization Formats: Several non-uniform formats target low-precision quantization. NF (Dettmers et al., 2023) places levels at the quantiles of a normal distribution, assuming normally distributed weights, and AF (Yoshida, 2023) refines this assumption. SF (Dotzel et al., 2024) captures heavier tails with a Student's t-distribution, and BOF (Blumenberg et al., 2026) optimizes levels per parameter block to adapt to local weight statistics. Despite differing derivations, all fix

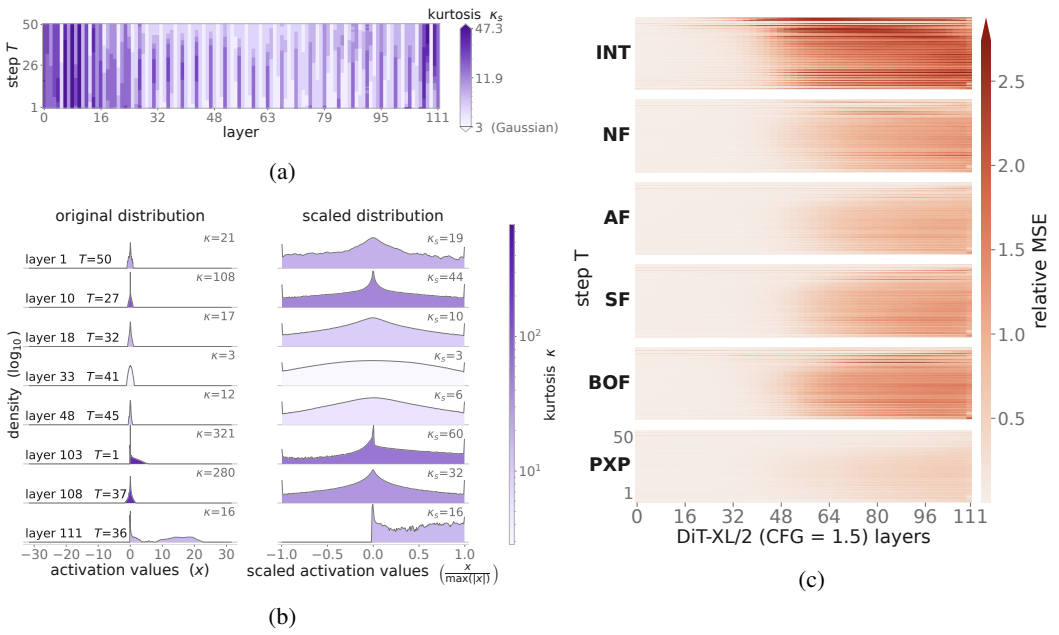


Figure 2: (a) Kurtosis, a measure of tail heaviness, of the group-wise scaled activations across layers and 50 denoising steps of DiT-XL/2 (CFG=1.5, group size=128). **(b)** Eight activation distributions of DiT-XL/2 sampled across layers and denoising steps, shown before and after group-wise scaling with group size 128. (c) Relative mean squared quantization error of existing formats relative to PXP, across layers and denoising steps of DiT-XL/2 (CFG 1.5).

level positions offline from a closed-form distributional assumption shared across the network. This breaks down for diffusion models, whose activation distributions vary across denoising steps and transformer blocks, so a codebook shaped for one regime is mismatched in another and the precision advantage of non-uniform formats largely vanishes.

3 Limitation of Conventional Non-Uniform Formats

Fig. 2a reports the kurtosis of activation distributions across layers and denoising steps of DiT-XL/2 (CFG 1.5). Kurtosis is the fourth standardized moment, $K = \mathbb{E}[(X-\mu)^4]/\sigma^4$, and measures tail heaviness, with $K = 3$ for any Gaussian. The observed values range from 3 to 47, so activations depart from the Gaussian assumption to widely varying degrees. Fig. 2b shows eight representative distributions before and after scaling, where K_s denotes the kurtosis after scaling. Scaling aligns their ranges but leaves these shape differences largely intact, so a codebook optimized for a single Gaussian-like prior, as existing non-uniform formats assume, cannot match activations across layers and denoising steps. These observations motivate adapting the level allocation itself to the target distribution. Fig. 2c reports the quantization error of each format across layers and 50 denoising steps of DiT-XL/2. For an activation tensor X and its quantized counterpart X_q, we measure the relative error $\rho = \frac{\|X - X_q\|_2^2}{\|X\|_2^2}$, where X is taken from the FP16 model. Every fixed format incurs large error, growing with depth in a manner consistent with accumulation through the network, whereas PXP stays low throughout. This shows that PXP's level allocation is what closes the gap.

4 Prefix-Point: Quantizer and Format

PXP places its representable values according to the observed input distribution rather than an assumed one. Each value consists of an exponent, which selects a range of magnitudes we call an exponent level, and a mantissa, which locates the value within that level. Encoding the exponent with a variable-length prefix code lets the mantissa width vary across levels, so harder-to-quantize levels

get more mantissa bits. A b-bit PXP format is thus specified by a prefix table over K exponent levels,

$$\mathcal{P}_K = \{(p_j, e_j)\}_{j=1}^{K}, \tag{1}$$

where $p_j \in \{0,1\}^{\ell_j}$ is the codeword assigned to exponent e_j, ℓ_j is its length and K is the number of exponent levels. Given $\mathcal{P}_K$, the PXP number consists of:

$$\boxed{\textbf{Sign } (s)\textbf{:}\ s \in \{0,1\},} \quad \boxed{\textbf{Prefix code } (p)\textbf{:}\ p \in (p_1, p_2, \ldots, p_K),} \quad \boxed{\textbf{Mantissa } (m)\textbf{:}\ m \in \mathbb{Z}_{\geq 0},} \tag{2}$$

where $\forall_{j=1}^{K} \mathcal{P}[p_j] = e_j$ and $0 \leq m < 2^{b-1-\ell}$ for prefix p with length ℓ. To quantize an input $x \in X$ to b-bit PXP format, we first normalize its magnitude to the range $(0, 1)$, while separating its sign:

$$\hat{x} = \left|\frac{x}{\alpha}\right| \quad \text{where} \quad \alpha = \max(|X|), \qquad s = \begin{cases} 0, & x \geq 0 \\ 1, & x < 0, \end{cases} \tag{3}$$

where α is the scale factor. We then assign $\hat{x}$ to an exponent level:

$$e_x = \text{clamp}(\lceil -\log_2(\hat{x}) \rceil, 0, K-1) \tag{4}$$

which is transmitted as its prefix code p_x, assigned by (1). Given length of prefix ℓ_x, mantissa receives the bits left over after the sign and the prefix, so its bitwidth b_m is:

$$b_m = b - \ell_x - 1. \tag{5}$$

and the mantissa is obtained as:

$$m_x = \begin{cases} \text{clamp}\left(\left\lfloor \hat{x} \cdot 2^{\,e_x + b_m} - 2^{b_m} \right\rceil,\ 0,\ 2^{b_m} - 1\right), & e_x \leq K-2, \\ \text{clamp}\left(\left\lfloor \hat{x} \cdot 2^{\,e_x + b_m - 1} \right\rceil,\ 0,\ 2^{b_m} - 1\right), & e_x = K-1. \end{cases} \tag{6}$$

The first case covers the normal levels and the second case is the subnormal level, which reuses the scale of the smallest normal level and also represents zero. A PXP number is therefore stored as (s, p, m) and the corresponding PXP quantizer is defined as:

$$\mathcal{Q}_{\text{PXP}}(x, \mathcal{P}_K, b) = \begin{cases} (-1)^s \times 2^{-e_x} \times \left(1 + \dfrac{m_x}{2^{b_m}}\right), & \text{if } e_x \leq K-2 \\ (-1)^s \times 2^{-e_x+1} \times \left(\dfrac{m_x}{2^{b_m}}\right), & \text{otherwise} \end{cases} \tag{7}$$

The dequantized value is recovered as $x_q = \alpha \cdot \mathcal{Q}_{\text{PXP}}(x, \mathcal{P}_K, b)$. Since the prefix and mantissa share a fixed b-bit budget, every bit spent on the prefix is taken from the mantissa. Levels contributing most to quantization error therefore receive short prefixes and finer resolution, while levels contributing little receive long prefixes and fewer values, freeing codewords for the levels that need them.

5 Optimizing the Prefix Codes for PXP

With the format and its quantizer established, the remaining question is how the prefix table should be chosen. Under a fixed per-value budget of b bits, the table $\mathcal{P}_K$ determines how those bits are divided between exponent and mantissa, and therefore where the representable values are placed. We select the table for which this placement incurs the smallest quantization error. For an input tensor $X = \{x_i\}_{i=1}^{N}$ the resulting problem is stated as follows:

$$\textbf{Decision Variables:}\ K \in \mathbb{Z}, \quad \mathcal{P}_K = \{(p_j, e_j)\}_{j=1}^{K} \text{ where } p_j \in \{0,1\}^{\ell_j} \tag{8}$$

$$\textbf{Objective:}\ \underset{K,\,\mathcal{P}_K}{\text{minimize}}\ \ \mathcal{E}(X, \mathcal{P}_K, b) = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \mathcal{Q}_{\text{PXP}}(x_i, \mathcal{P}_K, b) \cdot \alpha\right)^2 \tag{9}$$

$$\textbf{Constraints:}\ \textbf{1)}\ 1 \leq \ell_j \leq b-1 \ \text{ for } j < K, \qquad \textbf{2)}\ 1 \leq \ell_K \leq b-2, \tag{10}$$

$$\textbf{3)}\ \sum_{j=1}^{K} 2^{\,b-\ell^{(j)}-1} = 2^{\,b-1}, \qquad \textbf{4)}\ 2 \leq K \leq 2^{b-1} - 1. \tag{11}$$

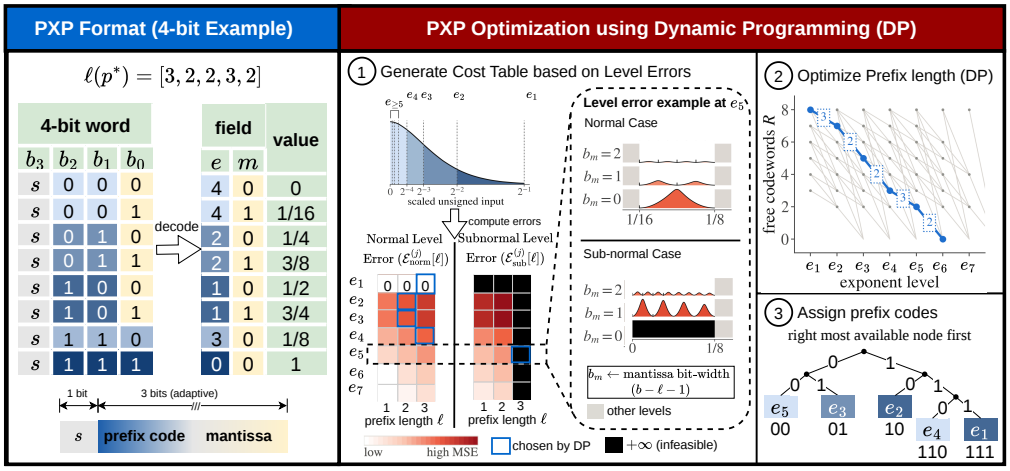


Figure 3: Overview of the PXP format and its optimization for a 4-bit example. **Left** Structure of a 4-bit PXP code. **Right** The three optimization stages, (1) building the cost table, which records the error of each exponent level under every prefix length, (2) selecting prefix lengths by dynamic programming over this table, and (3) assigning prefix codes by constructing the prefix tree.

Here, $\mathcal{E}(X, \mathcal{P}_K, b)$ is the MSE between X and its PXP-quantized counterpart. Constraints 1 and 2 keep every code within b bits, reserving at least one mantissa bit at the last level to encode zero. Constraint 3 is the Kraft equality $\sum_{j=1}^{K} 2^{-\ell_j} = 1$, ensuring no codeword is wasted, and constraint 4 bounds the number of levels based on b-bit budget. Exhaustive search over prefix tables is intractable, as the number of admissible prefix trees grows as $O\big((b-1)^{2^{b-1}}\big)$. Instead of enumerating trees, we treat the problem as a multiple-choice knapsack problem, and solve using dynamic programming. Fig. 3 illustrates a 4-bit PXP format (left) and the three stages of its optimization (right).

5.1 Building Cost Tables

Definition 1 (*Level Quantizer*)**.** Consider b-bit PXP quantization with K exponent levels, and let e_x denote the exponent level to which the input x is assigned. The level quantizer $\mathcal{Q}^{(j)}_{\text{PXP}}$ applies PXP quantization with prefix length ℓ_j to the values sharing the exponent e_j only:

$$\mathcal{Q}^{(j)}_{\text{PXP}}(x, \ell_j, K, b) = \begin{cases} (-1)^s \times 2^{-e_j} \times \left(1 + \dfrac{m}{2^{b_m}}\right), & e_x = e_j,\ e_j \le K-2, \\ (-1)^s \times 2^{-e_j+1} \times \dfrac{m}{2^{b_m}}, & e_x = e_j,\ e_j = K-1, \\ \dfrac{x}{\alpha}, & e_x \neq e_j, \end{cases} \tag{12}$$

where the mantissa bitwidth b_m follows from b and ℓ_j via (5).

Definition 2 (*Level Error*)**.** Let $\{x_i\}_{i=1}^{N}$ be the values to be quantized with scale α. For an exponent level e_j assigned prefix length ℓ_j, the level error is the quantization error contributed by the values that fall in that level:

$$\mathcal{E}^{(j)}[\ell_j] = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \alpha \cdot \mathcal{Q}^{(j)}_{\text{PXP}}(x_i, \ell_j, K, b)\right)^2, \tag{13}$$

which we categorize as the *normal level error* $\mathcal{E}^{(j)}_{\text{norm}}[\ell^{(j)}]$ when $K > e_j + 1$, and as the *subnormal level error* $\mathcal{E}^{(j)}_{\text{sub}}[\ell^{(j)}]$ when $K = e_j + 1$.

Lemma 1 (*Additivity of level-wise errors*)**.** *For any b-bit PXP format with K exponent levels and prefix lengths $\ell = (\ell_0, \ldots, \ell_{K-1})$, the objective in* (9) *decomposes into a sum of per-level terms,*

$$\mathcal{E}(X, \mathcal{P}_K, b) = \sum_{j=1}^{K} \mathcal{E}^{(j)}[\ell_j]. \tag{14}$$

Proof is in Appendix B. Minimizing (9) thus reduces to choosing K and ℓ that minimize the right-hand side of (14), which is a multiple-choice knapsack problem. Each level is assigned exactly one prefix length ℓ_j, costing $\mathcal{E}^{(j)}[\ell_j]$ and occupying $2^{b-\ell_j-1}$ codewords, such that all levels exactly fill the 2^{b-1} available codewords. We solve it by dynamic programming over precomputed level errors, stored in separate tables $\mathcal{E}^{(j)}_{\text{norm}}[\ell]$ and $\mathcal{E}^{(j)}_{\text{sub}}[\ell]$ since normal and subnormal levels incur different errors. With $K_{\max} = 2^{b-1} - 1$ from constraint 4 and constraints 1 and 2 in (10) admit $b-1$ prefix lengths for a normal level and $b-2$ for the subnormal one, the tables have sizes $(K_{\max}-1) \times (b-1)$ and $(K_{\max}-1) \times (b-2)$, where each entry is one evaluation of (13).

5.2 Prefix Length Optimization Using Dynamic Programming

Given the level errors $\mathcal{E}^{(j)}[\ell]$ for all $j \in \{1, \ldots, K_{\max}\}$ and admissible ℓ, we find the optimal prefix lengths $\ell^\star$ and level count $K^\star$ by dynamic programming. Let $\Lambda^{(j)}[M]$ denote the minimum accumulated error of the first j levels using exactly M codewords,

$$\Lambda^{(j)}[M] = \begin{cases} \min\limits_{1 \le \ell_j \le b-1} \left(\Lambda^{(j-1)}\left[M - 2^{\,b-\ell_j-1}\right] + \mathcal{E}^{(j)}_{norm}[\ell_j] \right), & \text{if } 0 < M \le 2^{\,b-1} - 2 \text{ and } 1 \le j < K_{max}, \\ \min\limits_{1 \le \ell_j \le b-2} \left(\Lambda^{(j-1)}\left[M - 2^{\,b-\ell_j-1}\right] + \mathcal{E}^{(j)}_{sub}[\ell_j] \right), & \text{if } M = 2^{\,b-1} \text{ and } 2 \le j \le K_{max}, \\ 0, & \text{if } M = 0 \text{ and } j = 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{15}$$

The first case extends the code with a normal level; the second closes it with the subnormal level, which occurs only at $M = 2^{\,b-1}$, where all available codewords have been consumed[1]; the third initializes the empty code; and the fourth assigns infinite cost to invalid entries, so that they are never selected by the minimization. Consequently $\Lambda[j]\left[2^{\,b-1}\right]$ is precisely the optimal error attainable by a complete format with exactly $K = j$ levels, and the optimum is obtained as:

$$K^\star = \operatorname*{arg\,min}_{2 \le j \le K_{max}} \Lambda^{(j)}\left[2^{\,b-1}\right], \qquad \mathcal{E}^\star(X, \mathcal{P}_{K^\star}, b) = \Lambda^{(K^\star)}\left[2^{\,b-1}\right], \tag{16}$$

with the optimal prefix length vector $\ell^\star$ recovered by backpointer tracing of $\Lambda^{(K^\star)}\left[2^{\,b-1}\right]$. Filling the table costs $\mathcal{O}\left(b\, K_{max} 2^{\,b-1}\right) = \mathcal{O}\left(b\, 2^{\,2b-2}\right)$ time. Because row j depends only on row $j-1$, all rows up to $j-2$ may be discarded once their terminal entry $\Lambda^{(j-2)}\left[2^{\,b-1}\right]$ has been retained as a candidate in (16), reducing the memory footprint to $\mathcal{O}\left(2^{\,b-1} + K_{max}\right)$.

Proposition 1 (*Dynamic programming optimality*)**.** *For every $j \in \{1, \ldots, K_{max}\}$ and codeword budget M, the value $\Lambda^{(j)}[M]$ computed by* (15) *equals the minimum achievable quantization error by the first j exponent levels using M codewords.*

Proof is in Appendix C. By Proposition 1, PXP therefore attains the MSE-optimal prefix assignment.

5.3 Assigning Prefix Codes

Since $\ell^\star$ satisfies the Kraft equality, it defines the leaf depths of a full binary prefix tree, which we construct in $\mathcal{O}(K^\star)$ time by placing levels in decreasing order of depth, each at the right-most free node of its depth (details in Appendix A).

5.4 Calibration-Driven Optimization

Optimizing prefix tables per input at runtime would add significant inference overhead (Sec. 6.4), so we calibrate them once per layer and denoising step on 64 calibration samples and keep them fixed. Per-layer and per-step codebooks add negligible storage overhead (Appendix K.1).

[1] At $M = 2^{\,b-1} - 1$ a single codeword remains. Treating the level as normal violates constraint 2, since the subnormal level that follows would need $\ell = b - 1$. Treating it as subnormal leaves that codeword unused, violating constraint 3. The state therefore admits no valid completion.

Table 1: Performance comparison of various quantization formats on DiT-XL/2 over ImageNet 256×256. Best result in each setting is shown in **bold**; the FP16 baseline is given in *grey*.

Model	Bit-width	Method	IS ↑	FID ↓	sFID ↓	Precision ↑	Recall ↑
	FP16 (Baseline)		*113.79*	*12.69*	*19.30*	*0.66*	*0.75*
DiT-XL/2 steps=100 CFG=1.0	W4A6	INT	2.36	229.16	455.98	0.02	0.02
		NF	21.95	70.67	212.63	0.29	0.54
		AF	26.99	74.07	52.69	0.27	0.62
		SF	13.13	117.77	69.36	0.17	0.48
		BOF	4.64	184.21	78.14	0.11	0.12
		PXP (Ours)	**85.73**	**21.57**	**22.28**	**0.58**	**0.73**
	FP16 (Baseline)		*111.39*	*14.19*	*19.27*	*0.66*	*0.74*
DiT-XL/2 steps=50 CFG=1.0	W4A6	INT	2.18	235.81	450.15	0.02	0.08
		NF	19.19	76.70	229.73	0.27	0.53
		AF	25.76	74.95	51.72	0.28	0.63
		SF	12.93	119.18	69.66	0.17	0.49
		BOF	4.49	184.66	78.66	0.12	0.11
		PXP (Ours)	**78.07**	**25.35**	**24.54**	**0.55**	**0.73**
	FP16 (Baseline)		*240.32*	*5.42*	*17.77*	*0.80*	*0.68*
DiT-XL/2 steps=100 CFG=1.5	W4A6	INT	2.81	211.32	422.86	0.03	0.12
		NF	64.72	39.30	151.76	0.48	0.57
		AF	74.35	39.02	44.49	0.44	**0.69**
		SF	33.22	78.52	63.00	0.26	0.62
		BOF	7.26	160.48	74.01	0.11	0.26
		PXP (Ours)	**191.30**	**8.62**	**21.68**	**0.75**	0.67
	FP16 (Baseline)		*236.11*	*5.40*	*17.57*	*0.81*	*0.67*
DiT-XL/2 steps=50 CFG=1.5	W4A6	INT	2.58	220.06	422.06	0.03	0.08
		NF	53.62	44.36	162.97	0.44	0.56
		AF	72.87	39.56	43.64	0.45	**0.70**
		SF	34.41	77.10	62.21	0.27	0.62
		BOF	7.53	160.27	74.08	0.11	0.27
		PXP (Ours)	**182.50**	**9.70**	**23.50**	**0.73**	0.67

Bit-width	Method	FID↓	sFID↓	IS↑	CLIP↑	IR↑
FP16 (Baseline)		*59.53*	*302.72*	*37.80*	*0.27*	*0.94*
W4A6	INT	345.511	363.46	2.40	0.13	-2.28
	NF	213.26	345.27	5.99	0.16	-2.11
	AF	85.18	294.37	26.79	0.25	-0.05
	SF	436.99	435.42	2.00	0.15	-2.27
	BOF	246.89	365.90	6.16	0.20	-1.65
	PXP (Ours)	**69.75**	**290.37**	**32.99**	**0.26**	**0.56**

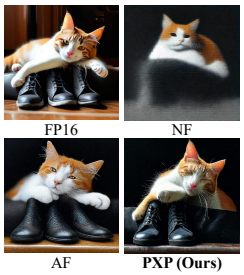


Figure 4: **(Left)** Text-to-image generation performance of the PixArt-Σ model on the COCO dataset. Best result in each setting is shown in **bold**; the FP16 baseline is given in *grey*. **(Right)** Qualitative comparison of images generated by PixArt-Σ model on the COCO dataset under W4A6 quantization.

6 Experiments

6.1 Experimental Setup

We evaluate PXP on class-conditional generation with DiT-XL/2 (Peebles & Xie, 2023) on ImageNet 256 × 256 (Deng et al., 2009), text-to-image generation with PixArt-Σ (Chen et al., 2024) on COCO (Lin et al., 2014), and video generation with Open-Sora (Zheng et al., 2024) on VBench (Huang et al., 2024). We report FID (Heusel et al., 2017), sFID, IS (Salimans et al., 2016), precision and recall, adding CLIPScore (Hessel et al., 2021) and ImageReward (Xu et al., 2023) for text-to-image and eight VBench dimensions for video following Xu et al. (2026). We compare against INT, NF, AF, SF and BOF, all using absolute-max scaling without equivalent transformations or advanced calibration to isolate the format itself. Sampling and quantization details are in Appendix D.

Table 2: Performance comparison of various quantization formats on OPEN-SORA V1.2 over VBench. Best results are shown in **bold**. For dynamic degree, best means nearest to FP16, since flicker inflates it. FP16 baseline is given in *grey*. †Collapsed, with overall consistency below half of FP16.

Bit-width	Method	Subject Consist.	BG. Consist.	Motion Smooth.	Dynamic Degree	Aesthetic Quality	Imaging Quality	Scene Consist.	Overall Consist.
FP16 (Baseline)		*90.73*	*97.65*	*98.00*	*71.94*	*54.06*	*55.29*	*40.60*	*26.31*
W4A4	INT†	87.87	94.91	93.45	99.44	33.76	42.56	0.00	3.54
	NF†	86.51	94.66	94.68	90.56	33.44	41.10	0.57	8.65
	AF	**85.57**	95.90	**97.21**	58.33	42.71	43.84	21.37	19.97
	SF†	85.53	95.04	96.70	80.83	34.39	39.02	2.83	10.54
	BOF†	69.82	90.30	92.59	100.00	30.16	37.18	2.28	9.34
	PXP (Ours)	85.20	**95.99**	96.91	**70.83**	**45.55**	**48.60**	**33.14**	**23.00**

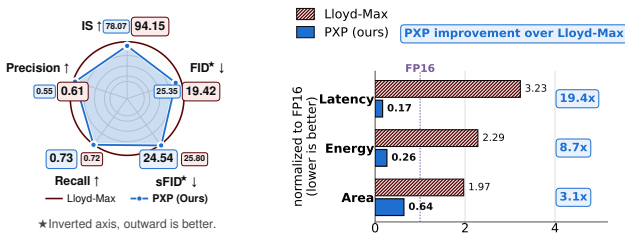


Figure 5: Comparison with a Lloyd-Max calibrated codebook (MSE-optimal) at W4A6. **(Left)** Generation quality on DiT-XL/2 (CFG=1.0, 50 steps). **(Right)** Latency, energy and area of Lloyd-Max and PXP, normalized to the FP16 baseline.

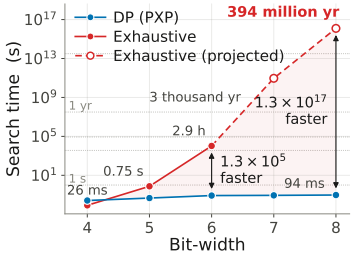


Figure 6: Search time comparison of exhaustive search vs. our dynamic programming (DP) based PXP optimization.

6.2 COMPARISON WITH EXISTING NON-UNIFORM FORMATS

Table 1 compares all formats at W4A6 on DiT-XL/2 over ImageNet 256×256 in four configurations (other bit-widths in Appendix H). At 100 steps with CFG $= 1.5$, PXP reaches an FID of 8.62 against 5.42 for FP16, while the strongest baseline, AF, degrades to 39.02, a $4.5\times$ gap. The remaining formats fail distinctly, with NF destroying spatial layout (sFID up to 229.73), SF losing per-image fidelity (FID > 77), and INT collapsing entirely (FID > 211). Baseline precision drops to 0.02 to 0.48 while recall degrades far less, indicating unrealistic rather than merely less diverse samples. PXP retains 70 to 80% of full-precision IS and 83 to 94% of its precision, and is the only format within 6 sFID of FP16 in every setting (see Table 1; qualitative results in Appendix K.2).

6.3 GENERALIZATION ACROSS TASKS AND ARCHITECTURES

To test generalization, we evaluate PixArt-Σ on COCO (Fig. 4) and OPEN-SORA V1.2 on VBench (Table 2). On COCO, the ImageNet ordering persists, with PXP closest to FP16 and AF the only competitive baseline. Prompt alignment separates the formats more sharply than image quality, as most baselines lose the conditioning altogether, while PXP nearly matches FP16 in CLIP score and is the only format with positive ImageReward. The samples in Fig. 4(right) reflect this, with AF losing fine detail and NF blurring. Video generation at W4A4 is more demanding, since errors must remain consistent across frames as well as within them. INT, NF, SF and BOF collapse, and their inflated dynamic degree reflects frame-to-frame flicker rather than motion, so this metric is informative only relative to FP16. AF avoids collapse but loses about half of the scene consistency. PXP stays closest to FP16 on the dimensions most sensitive to quantization, namely scene consistency, overall consistency and dynamic degree, while remaining comparable on subject consistency and motion smoothness, making it the only format that preserves both temporal stability and prompt grounding. Qualitative comparisons for both tasks are in Appendix K.2.

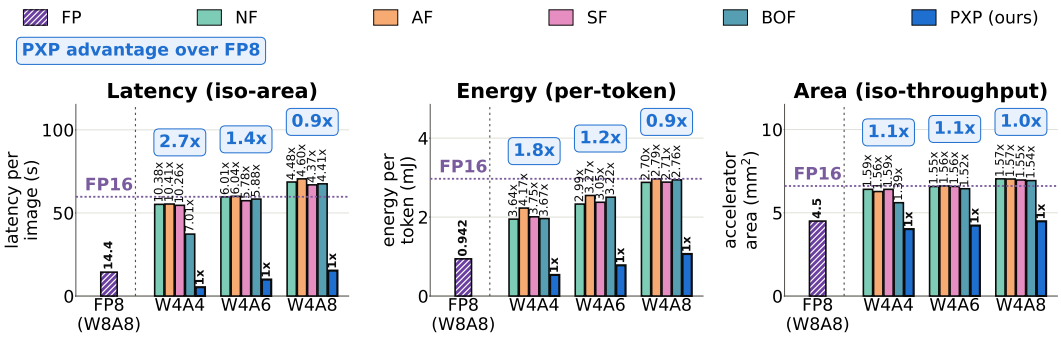


Figure 7: Hardware cost of existing formats vs. PXP on DiT-XL/2 (50 denoising steps), covering the activation quantizer and MAC unit. Latency is measured at iso-area, energy per token, and area at iso-throughput. Ratios above bars denote PXP's advantage over each format.

6.4 Analysis

Comparison with the Lloyd-Max Optimal Codebook: Fig. 5 compares PXP at W4A6 on DiT-XL/2 (CFG=1.0, 50 steps) with a Lloyd-Max codebook fitted by k-means on calibration activations, which minimizes per-tensor MSE. PXP trades some quality for efficiency, trailing Lloyd-Max on FID and IS while matching it on sFID and recall (Fig. 5, left). In exchange, it achieves $19.4\times$, $8.7\times$ and $3.1\times$ lower latency, energy and area (Fig. 5, right), since an unconstrained codebook requires full-LUT decoding and a full-precision datapath, whereas PXP decodes only the exponents to a narrow floating-point operand. PXP thus approaches MSE-optimal quality at a fraction of the hardware cost.

Exhaustive Search vs. Dynamic Programming: Fig. 6 compares exhaustive search with our dynamic programming (DP) on a single A100. The DP solves each tensor in under a second at every bit-width, whereas exhaustive search takes hours per tensor at 6 bits, prohibitive across all layers and denoising steps, and a projected three thousand years at 7 bits. The DP is thus $1.3 \times 10^5\times$ faster at 6 bits, making prefix table optimization practical.

Hardware Efficiency: We implement the quantizer and MAC unit of each format in RTL and measure latency, energy and area on real tensors (Fig. 7, details in Appendix J). Latency is compared at iso-area and area at iso-throughput, so that each metric reflects how efficiently a format uses silicon, while energy is reported per token to remain independent of array size. Against existing non-uniform formats, PXP is up to $10\times$ faster, $4.2\times$ more energy efficient and $1.6\times$ smaller at W4A4, with the gap narrowing to about $4.5\times$ and $2.7\times$ in latency and energy at W4A8. Two effects drive this. The decoder stores one entry per exponent level rather than per codeword, and, more importantly, PXP codes decode into narrow floating-point operands, whereas LUT formats may hold arbitrary values and require a full-precision datapath. PXP is also $2.7\times$ and $1.4\times$ faster than FP8 at W4A4 and W4A6, respectively, and comparable ($0.9\times$) at W4A8.

7 Conclusion

Activation distributions in diffusion models vary in shape, not only in scale, across layers and denoising steps, and non-uniform formats with levels fixed at design time cannot follow this variation. We proposed PXP, a format that allocates exponent levels and per-level mantissa widths to fit the distribution of each layer and denoising step, while retaining a floating-point structure that keeps decoding and arithmetic cheap in hardware. We showed that the quantization error separates across exponent levels, which turns the allocation into a multiple-choice knapsack problem solved exactly by dynamic programming in under a second per tensor, more than $10^5\times$ faster than exhaustive search and entirely offline. Across class-conditional, text-to-image and video diffusion, PXP consistently outperforms existing non-uniform formats, lowering FID by up to $4.5\times$ over the strongest baseline while being up to $6\times$ faster at W4A6. These results identify the numeric format itself as a key design dimension for low-bit diffusion inference.

AI USE STATEMENT

In this work, we used generative AI tools to implement the hardware evaluation, and assist in refining and checking proofs. Specifically, LLMs assisted in writing the RTL for the quantizer and multiply-accumulate units of each format used in Section 6.4. The proofs of Lemma 1, Proposition 1 and Proposition 2 were devised by the authors, and LLMs were used only to refine their presentation and check them for errors. We have not used generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design research methodology or experiments, or interpret results, and the remaining required disclosure tasks, namely synthetic data generation, translation and dataset cleaning, are not applicable to this work. Additionally, we used generative AI tools for proofreading and editing the paper to improve readability. We have reviewed all AI-assisted work. The RTL implementations were verified by two authors, including bit-exact comparison against the software quantizer (Appendix J). All edited text was checked by the authors against the intended meaning and the reported results, and all proofs were verified by the authors after refinement. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Anonymized source code for the quantizer, DP solver, evaluation pipeline and RTL designs is provided in the supplementary material.

REFERENCES

Rajeev Balasubramonian, Andrew B Kahng, Naveen Muralimanohar, Ali Shafiee, and Vaishnav Srinivas. Cacti 7: New tools for interconnect exploration in innovative off-chip memories. *ACM Transactions on Architecture and Code Optimization (TACO)*, 14(2):1–25, 2017.

Patrick Blumenberg, Thomas Graave, and Tim Fingscheidt. Improving block-wise llm quantization by 4-bit block-wise optimal float (bof4): Analysis and variations. In *International Conference on Learning Representations*, volume 2026, pp. 6796–6827, 2026.

Robert Brayton and Alan Mishchenko. Abc: An academic industrial-strength verification tool. In *International Conference on Computer Aided Verification*, pp. 24–40. Springer, 2010.

Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *European Conference on Computer Vision*, pp. 74–91. Springer, 2024.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-dit: Accurate post-training quantization for diffusion transformers. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28306–28315. IEEE, 2025.

James Cherry. OpenSTA: Open-source static timing analyzer. `https://github.com/parallaxsw/OpenSTA`, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Jordan Dotzel, Yuzong Chen, Bahaa Kotb, Sushma Prasad, Gang Wu, Sheng Li, Mohamed S Abdelfattah, and Zhiru Zhang. Learning from students: Applying t-distributions to explore accurate and efficient formats for LLMs. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=iJlPJsTw2B`.

Aleksandr Dremov, David Grangier, Angelos Katharopoulos, and Awni Hannun. Compute-optimal quantization-aware training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=QpbtT95S95`.

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. URL https://openreview.net/forum?id=rkgO66VKDS.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Andrey Kuzmin, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, and Tijmen Blankevoort. Fp8 quantization: The power of the exponent. *Advances in Neural Information Processing Systems*, 35:14651–14662, 2022.

Dongyue Li, Zechun Liu, Kai Yi, Changsheng Zhao, Raghuraman Krishnamoorthi, Harshit Khaitan, Hongyang R. Zhang, and Steven Li. Winiq: Accelerating quantization-aware training of LLMs around saddle points, 2026. URL https://openreview.net/forum?id=IEGwgxe13Y.

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. URL https://openreview.net/forum?id=POWv6hDd9XH.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Peter Markstein. The new ieee-754 standard for floating point arithmetic. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Cuong Pham, Dung Anh Hoang, Cuong C. Nguyen, Trung Le, Dinh Phung, Gustavo Carneiro, and Thanh-Toan Do. Beyond uniformity: Sample and frequency meta weighting for post-training quantization of diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FDdOD3qwS7.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1972–1981. IEEE, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys-a free verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, volume 97, pp. 1–6, 2013.

Junyi Wu, Haoxuan Wang, Yuzhang Shang, Mubarak Shah, and Yan Yan. Ptq4dit: Post-training quantization for diffusion transformers. *Advances in neural information processing systems*, 37: 62732–62755, 2024.

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*, pp. 38087–38099. PMLR, 2023.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Qinkai Xu, Lin Yang, Li Li, Yuxiang Fu, et al. Veta-dit: Variance-equalized and temporally adaptive quantization for efficient 4-bit diffusion transformers. *Advances in Neural Information Processing Systems*, 38:152907–152938, 2026.

Davis Yoshida. Nf4 isn't information theoretically optimal (and that's good). *arXiv preprint arXiv:2306.06965*, 2023.

Qian Zeng, Jie Song, Yuanyu Wan, Huiqiong Wang, and Mingli Song. Sampling-aware quantization for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 35831–35840, June 2026.

Jiaji Zhang, Ruichao Sun, Hailiang Zhao, Jiaju Wu, Peng Chen, Hao Li, Yuying Liu, Kingsum Chow, Gang Xiong, and Shuiguang Deng. Segquant: A semantics-aware and generalizable quantization framework for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43717–43726, June 2026a.

Peiyuan Zhang, Matthew Noto, Wenxuan Tan, Chengquan Jiang, Will Lin, Wei Zhou, and Hao Zhang. Attn-QAT: 4-bit attention with quantization-aware training. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=pqOlyvsmU3.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

APPENDIX

A DETAILS ON PREFIX CODE ASSIGNMENT

The DP in (15) returns only the prefix length $\ell_j^\star$ of each exponent level. This section describes how the prefix table $\mathcal{P}_{K^\star}$ is constructed from these lengths and shows that it defines a complete prefix code.

Excluding the sign bit, a b-bit PXP word indexes one of 2^{b-1} codewords, which we view as the leaves of a complete binary tree of depth $b-1$. A node at depth ℓ is identified by the ℓ-bit path from the root, and its subtree contains the $c = 2^{b-\ell-1}$ codewords that share this path as their leading bits. Assigning an exponent level the prefix length ℓ therefore amounts to assigning it a node at depth ℓ, whose path becomes the level's prefix and whose subtree holds the level's $2^{b-\ell-1}$ mantissa values. The prefix table is complete when the assigned nodes have disjoint subtrees that together cover every leaf, which is possible exactly when the lengths satisfy the Kraft equality. In terms of codeword indices, each assigned subtree is a contiguous block of size c starting at a multiple of c.

We order the levels as $j_1, \dots, j_{K^\star}$ by decreasing prefix length, that is, by increasing subtree size $c_k = 2^{b-\ell_{j_k}^\star - 1}$, breaking ties by exponent index. Each level is assigned the right-most free node at its depth. Starting from $C^{(0)} = 2^{b-1}$, this places the subtrees from the top of the codebook downward,

$$C^{(k)} = C^{(k-1)} - c_k, \qquad k = 1, \dots, K^\star, \tag{A1}$$

so that level j_k occupies the codewords $\big[C^{(k)},\, C^{(k-1)}\big)$ and $C^{(k)}$ counts the leaves still free. Its prefix is the path to the root of this subtree, given by the leading $\ell_{j_k}^\star$ bits of its first codeword,

$$p_{j_k} = \mathrm{bin}_{\ell_{j_k}^\star}\Big(C^{(k)} \gg \big(b - \ell_{j_k}^\star - 1\big)\Big), \tag{A2}$$

where $\mathrm{bin}_\ell(\cdot)$ denotes the ℓ-bit binary representation. The resulting prefix table is $\mathcal{P}_{K^\star} = \{(p_j, e_j)\}_{j=1}^{K^\star}$, and Algorithm 1 summarizes the procedure.

Algorithm 1 Prefix table construction. Input is the optimal prefix lengths $\ell^\star$ and bit-width b, output is the prefix table $\mathcal{P}_{K^\star}$.

1: Sort levels by decreasing $\ell_j^\star$, ties by exponent index, giving $j_1, \dots, j_{K^\star}$
2: $C \leftarrow 2^{b-1}$ ▷ number of free leaves
3: **for** $k = 1$ to $K^\star$ **do**
4: $\quad c \leftarrow 2^{\,b-\ell_{j_k}^\star - 1}$ ▷ subtree size of level j_k
5: $\quad C \leftarrow C - c$ ▷ right-most free node, leaves $[C,\, C+c)$
6: $\quad p_{j_k} \leftarrow \mathrm{bin}_{\ell_{j_k}^\star}\big(C \gg (b - \ell_{j_k}^\star - 1)\big)$ ▷ path to the node
7: **end for**
8: **return** $\mathcal{P}_{K^\star} = \{(p_j, e_j)\}_{j=1}^{K^\star}$

Proposition 2 (Validity of the construction)**.** *If $\ell^\star$ satisfies the Kraft equality $\sum_j 2^{-\ell_j^\star} = 1$, Algorithm 1 produces a prefix table $\mathcal{P}_{K^\star}$ that defines a complete prefix code.*

Proof. Multiplying the Kraft equality by 2^{b-1} gives $\sum_k c_k = 2^{b-1}$, so before level j_k is placed the number of free leaves is $C^{(k-1)} = \sum_{i \ge k} c_i$. Every c_i with $i \ge k$ is a power of two no smaller than c_k and hence a multiple of it, so $C^{(k-1)}$ and $C^{(k)} = C^{(k-1)} - c_k$ are both multiples of c_k. The leaves $\big[C^{(k)}, C^{(k-1)}\big)$ therefore form exactly the subtree of a node at depth $\ell_{j_k}^\star$, namely the right-most free node at that depth, whose path from the root is the prefix p_{j_k} in (A2). These subtrees are disjoint by construction, so no assigned node is an ancestor of another and no prefix in $\mathcal{P}_{K^\star}$ is a prefix of another. Since $C^{(K^\star)} = 2^{b-1} - \sum_k c_k = 0$, the subtrees also cover every leaf, so no codeword is left unused and the prefix tree is full. The code is thus prefix-free and complete. Sorting dominates the cost, and since $\ell_j^\star$ takes at most $b-1$ values, a counting sort gives $\mathcal{O}(K^\star)$ time. □

Example. For the 4-bit format of Fig. 3, $\ell^\star = (3, 2, 2, 3, 2)$ for $e_1, \ldots, e_5$, giving block sizes $(1, 2, 2, 1, 2)$ over a codebook of $2^3 = 8$ codewords. The order is e_1, e_4, e_2, e_3, e_5, and (A1) gives $C = 7, 6, 4, 2, 0$. The resulting prefixes are 111 for e_1, 110 for e_4, 10 for e_2, 01 for e_3 and 00 for e_5, matching the figure.

B Proof of Lemma 1

We prove Lemma 1 stating that the total error $\mathcal{E}(X, \mathcal{P}_K, b)$ of a tensor X under a b-bit format with K levels is the sum of per-level errors $\mathcal{E}^{(j)}$, where level j depends only on the values falling into it and on its own prefix length ℓ_j. Levels do not interact, which is what makes dynamic programming applicable.

Proof. Every x_i is assigned exactly one exponent level by (4), so the index sets $S_j = \{\, i \mid e_{x_i} = e_j \,\}$ are pairwise disjoint and cover $\{1, \ldots, N\}$.

By the third case of (12), $\alpha\, \mathcal{Q}^{(j)}_{\text{PXP}}(x_i, \ell_j, K, b) = x_i$ for every $i \notin S_j$, so these terms vanish in (13) and the level error reduces to a sum over S_j alone. For $i \in S_j$ the first two cases of (12) coincide with the full quantizer of (7).

Therefore, summing over level-wise errors,

$$\sum_{j=0}^{K-1} \mathcal{E}^{(j)}[\ell_j] = \frac{1}{N} \sum_{j=0}^{K-1} \sum_{i \in S_j} \left(x_i - \alpha \cdot \mathcal{Q}^{(j)}_{\text{PXP}}(x_i, \ell_j, K, b)\right)^2 \tag{A3}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \alpha \cdot \mathcal{Q}^{(j)}_{\text{PXP}}(x_i, \ell_j, K, b)\right)^2 \tag{A4}$$

$$= \mathcal{E}(X, \mathcal{P}_K, b). \tag{A5}$$

□

C Proof of Proposition 1

We prove Proposition 1 stating that the table entry $\Lambda^{(j)}[M]$ filled in by the recursion in (15) is exactly the smallest error achievable by the first j of the K_{max} exponent levels when at most M codewords are spent on them. The solution returned by the recursion is therefore globally optimal.

Proof. We prove the claim by induction on j, that $\Lambda^{(j)}[M]$ is the least error of any assignment of the first j levels using exactly M codewords, and $+\infty$ if no such assignment exists.

Base case. For $j = 0$ the only assignment is the empty one, which uses no codewords and incurs no error. Hence $\Lambda^{(0)}[0] = 0$ and $\Lambda^{(0)}[M] = +\infty$ otherwise, so the claim holds.

Inductive step. Assume $\Lambda^{(j-1)}$ is optimal at every budget. Assigning level j a prefix length ℓ_j costs $\mathcal{E}^{(j)}[\ell_j]$ and consumes $2^{\,b-\ell_j-1}$ codewords, leaving $M - 2^{\,b-\ell_j-1}$ for the preceding levels, whose least error is $\Lambda^{(j-1)}\big[M - 2^{\,b-\ell_j-1}\big]$ by assumption. Every valid assignment at (j, M) arises from exactly one such choice of ℓ_j, and (15) evaluates the resulting cost for all admissible ℓ_j and keeps the smallest. Hence $\Lambda^{(j)}[M]$ is optimal. Here level j is normal while codewords remain ($M \le 2^{b-1} - 2$) and subnormal once the codebook is exhausted ($M = 2^{b-1}$), as in (15). All other states are invalid, hence $+\infty$. The claim holds at $j = 0$ and is preserved from $j - 1$ to j, so it holds for all j. In particular $\Lambda^{(j)}\big[2^{b-1}\big]$ is optimal for $K = j$, and (16) returns the global optimum. □

D Experimental Details

We quantize the pretrained DiT-XL/2 (Peebles & Xie, 2023) released by the original authors and generate 256×256 images for ImageNet (Deng et al., 2009). Sampling uses the DDIM solver (Song et al., 2020) with 50 and 100 steps and classifier-free guidance (CFG) scales of 1.0 and 1.5, giving

the four configurations in Table 1. For each configuration we generate 10,000 images. We quantize PixArt-Σ (Chen et al., 2024) at W4A6 and generate 1,024 COCO (Lin et al., 2014) images, using a 20-step DPM-solver (Lu et al., 2022) with CFG of 4.5. We use CLIPScore (Hessel et al., 2021), which captures how well each image matches its prompt and ImageReward (Xu et al., 2023), which approximates human preference on complex prompts. We quantize STDiT3, the denoising backbone of OPEN-SORA V1.2 (Zheng et al., 2024), and generate videos on VBench (Huang et al., 2024), using 50-step DDIM with CFG of 4.0. Following Xu et al. (2026), we report eight VBench (Huang et al., 2024) dimensions, namely subject consistency, background consistency, motion smoothness, dynamic degree, aesthetic quality, imaging quality, scene consistency and overall consistency. Video results are reported at W4A4.

All formats use symmetric absolute-max scaling without equivalent transformations, rotations or learned clipping, so that differences reflect the format alone. On ImageNet and Open-Sora, weights and activations are scaled group-wise with group size 128. On COCO, activations are scaled per token and weights per channel. INT uses symmetric uniform levels. NF (Dettmers et al., 2023), AF (Yoshida, 2023), SF (Dotzel et al., 2024) and BOF (Blumenberg et al., 2026) are implemented following the original papers with their official code and default hyperparameters, with codebooks generated at each target bit-width. BOF is applied to activations with the same group size of 128 used for scaling. Each baseline uses a single codebook shared across all layers and denoising steps.

E CODEBOOK DIVERSITY ACROSS LAYERS AND TIMESTEPS

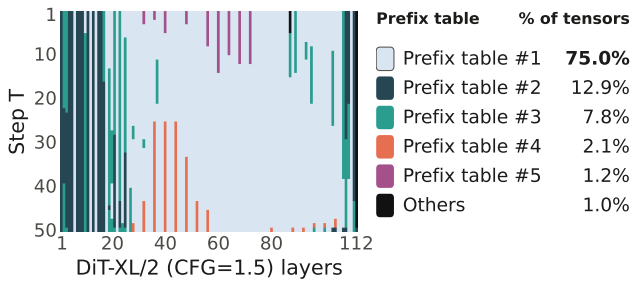


Figure A1: Map of calibrated codebooks across layers and denoising steps of DiT-XL/2 (CFG=1.5) across 50 steps.

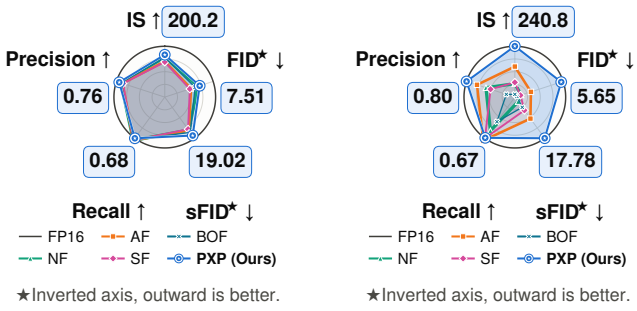
Fig. A1 shows how the selected codebook varies across layers and denoising steps. Most middle layers converge on a single codebook, covering 75% of all activations, whereas the early and final layers vary along both axes, so no single codebook is optimal for the whole network. Allowing a separate codebook per layer and per step is nonetheless cheap, since Fig. A5 shows the resulting table storage is negligible relative to the model size.

F DISENTANGLING WEIGHT AND ACTIVATION QUANTIZATION

To locate PXP's gain, we quantize weights and activations separately for DiT-XL/2 (CFG=1.5, steps=100). With weights only (Fig. A2a), all formats perform well and PXP improves only slightly, as weights are static after training and a fixed level set fits them regardless of input. With activations only (Fig. A2b), which vary across layers and denoising steps, baselines degrade sharply while PXP stays close to FP16 (detailed results in Table A1). PXP's gain is therefore driven almost entirely by activations.

G CALIBRATED VS. RUNTIME-OPTIMIZED PREFIX TABLE

We analyze what is lost by fixing the prefix table at calibration instead of re-optimizing it per input on DiT-XL/2 (CFG=1.5, steps=50). Runtime optimization is $110\times$ slower on an A100 (Fig. A3, top) yet gains little, as it selects the calibrated table for 95.1% of activations and stays near-optimal in MSE



(a) Weight-only (4-bit). (b) Activation-only (6-bit).

Figure A2: Weight-only and activation-only quantization of existing formats and PXP on DiT-XL/2 over ImageNet 256×256 (CFG=1.5, 100 steps). **(a)** Weight-only 4-bit quantization comparison with FP16 activations (W4A16). **(b)** Activation-only 6-bit quantization comparison with FP16 weights (W16A6).

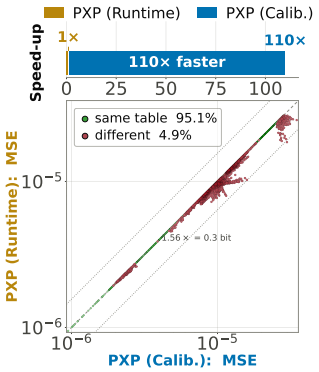
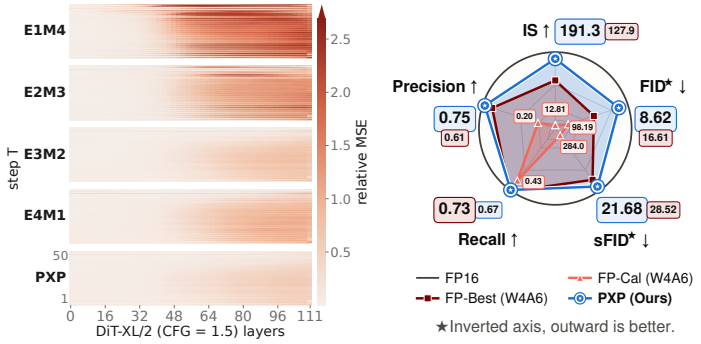


Figure A3: Calibrated table vs. runtime optimization. **(Top)** Speedup. **(Bottom)** 6-bit quantization MSE (diagonal marks same table selection).

for the rest (Fig. A3, bottom). Fixing the table at calibration thus preserves accuracy while keeping optimization off the inference path.

H IMAGENET RESULTS ACROSS MORE BIT-WIDTHS

Table A1 compares PXP with existing non-uniform formats on DiT-XL/2 over ImageNet 256×256, using 100 denoising steps with CFG = 1.5 and 10,000 generated samples, holding weights at 4 bits while sweeping activations from 5 to 8 bits. The advantage of PXP is largest at low precision. At W4A5 every baseline collapses, with FID above 190 and IS below 4, whereas PXP still reaches 9.35 FID and 180.73 IS. At W4A6 the strongest baseline (AF) reaches 39.02 FID, which PXP improves by 4.5× to 8.62. The gap narrows as precision increases. At W4A7 only NF remains competitive with PXP, the two outperforming the remaining formats by a wide margin, and at W4A8 all formats are close with no clear winner. PXP is therefore robust across bit-widths and is particularly advantageous below 7 bits, where existing formats degrade sharply. Table A1 also reports weight-only (W4A16) and activation-only (W16A6) quantization, which are visualized in Fig. A2 and discussed in Appendix F.



(a) (b)

Figure A4: Comparison with conventional low-precision floating point (FP) formats. (a) Relative MSE of 6-bit activation quantization on DiT-XL/2 (CFG=1.5) over 50 denoising steps, for standard FP formats and PXP. (b) ImageNet 256 × 256 generation with DiT-XL/2 (CFG=1.5, $T = 100$) at W4A6, where FP-Cal calibrates the exponent width and bias per layer and per step, and FP-Best is the best fixed ExMy format under 4-bit weights and 6-bit activations selected by generation quality.

Table A1: Performance across various bit-width settings for weight and activation quantization on DiT-XL/2 over ImageNet 256×256, with 100 sampling steps and CFG = 1.5. Best result at each bit-width is shown in **bold**; the FP16 baseline is given in *grey*.

Bit-width	Method	IS ↑	FID ↓	sFID ↓	Precision ↑	Recall ↑
FP16 (Baseline)		*240.32*	*5.42*	*17.77*	*0.80*	*0.68*
W4A5	NF	3.37	197.46	433.88	0.04	0.14
	AF	3.21	197.72	420.30	0.04	0.14
	SF	3.37	190.63	428.12	0.05	0.16
	BOF	1.45	349.60	319.81	0.00	0.00
	PXP (Ours)	**180.73**	**9.35**	**21.04**	**0.73**	**0.68**
W4A6	NF	64.72	39.30	151.76	0.48	0.57
	AF	74.35	39.02	44.49	0.44	**0.69**
	SF	33.22	78.52	63.00	0.26	0.62
	BOF	7.26	160.48	74.01	0.11	0.26
	PXP (Ours)	**191.30**	**8.62**	**21.68**	**0.75**	0.67
W4A7	NF	**204.15**	**6.80**	**18.92**	**0.76**	0.70
	AF	132.06	18.01	62.66	0.65	0.62
	SF	107.68	23.52	36.58	0.56	**0.72**
	BOF	155.41	12.35	26.74	0.67	**0.72**
	PXP (Ours)	198.24	7.66	19.25	**0.76**	0.68
W4A8	NF	196.22	**7.27**	20.71	0.74	0.70
	AF	191.78	7.34	**18.29**	0.75	0.70
	SF	153.17	12.48	25.16	0.66	**0.72**
	BOF	185.42	8.84	20.96	0.72	0.70
	PXP (Ours)	**199.55**	7.54	19.02	**0.76**	0.68
W4A16	NF	189.21	7.88	22.20	0.73	**0.71**
	AF	169.26	9.99	21.98	0.70	**0.71**
	SF	166.18	10.47	22.78	0.69	**0.71**
	BOF	186.53	8.64	20.75	0.73	0.70
	PXP (Ours)	**200.20**	**7.51**	**19.02**	**0.76**	0.68
W16A6	NF	63.56	40.92	160.83	0.48	0.55
	AF	144.39	16.27	34.02	0.63	**0.71**
	SF	69.70	46.47	55.29	0.40	0.67
	BOF	13.12	130.82	71.64	0.14	0.40
	PXP (Ours)	**240.76**	**5.65**	**17.78**	**0.80**	0.67

I Comparison with Floating-Point Formats

Given the resemblance of PXP to floating point, minifloats (Kuzmin et al., 2022) are its closest alternative, since they also place levels geometrically but fix their exponent and mantissa widths at design time. In Fig. A4a, which reports the relative MSE of 6-bit activation quantization on DiT-XL/2 (CFG=1.5) over 50 denoising steps following the methodology in Section 3, E3M2 is the best minifloat yet still incurs significant error than PXP.

This carries over to generation quality. Fig. A4b reports W4A6 results on ImageNet 256×256 relative to FP16, where FP-Best is the best fixed minifloat by generation quality (E2M1 weights, E3M2 activations) and FP-Cal selects the exponent width and bias per tensor by MSE calibration. PXP is best on IS, FID, sFID and Precision (191.3, 8.62, 21.68, 0.75 against 127.9, 16.61, 28.52, 0.61) and trails only slightly in Recall (0.67 against 0.73). Calibration does not close the gap and FP-Cal falls behind even the fixed format, since per-tensor MSE favors mantissa precision over dynamic range and selects E2M3 for roughly 89% of activation layers where E3M2 is what preserves quality. Tuning format parameters by error alone is therefore insufficient, and the gain of PXP comes from reallocating resolution within the format.

J Hardware Evaluation Details

We write synthesizable RTL Verilog for the two units whose cost depends on the format, namely the activation quantizer and the multiply accumulate (MAC) unit, verify the functionality against the software quantizer, and push the designs through an open-source synthesis flow with identical

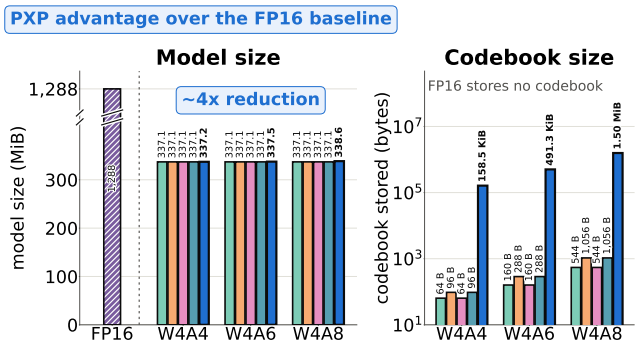


Figure A5: Model and codebook size.

constraints and identical input data. Designs are synthesized with Yosys (Wolf et al., 2013) and ABC (Brayton & Mishchenko, 2010) to the NanGate 45 nm library at the *typical* corner, timed register to register in OpenSTA (Cherry, 2023), and power-estimated using gate-level switching activity captured on real DiT-XL/2 (CFG=1.5) tensors with the Icarus Verilog simulator. Per-token and per-group scale multiplications are shared by all formats and lie outside the measured units, and since every format stores 4-bit weight codes and moves n_a-bit activation codes, buffers and off-chip traffic are constant across the comparison. Because a PXP code carries an explicit exponent and mantissa, its product is a narrow mantissa multiply (3×3, 5×3 and 7×3 bits at W4A4, W4A6 and W4A8, including the leading one, versus 11×11 for FP16) shifted by the exponent sum. A group of 128 products then accumulates exactly in a 19-, 23- or 25-bit fixed-point register. A look-up table (LUT) on the other hand, may hold an arbitrary value and must therefore decode into an FP16 operand on an IEEE FP16 multiply accumulate. This applies to the Lloyd-Max codebook as well, which we implement as a programmable LUT decoding to FP16, since its levels are unconstrained and refitted per tensor. Each baseline is given the cheapest of four decoder architectures with its tables hardwired into logic, whereas PXP remains programmable per layer, so the setup favors the baselines throughout. Unit measurements are then scaled to an output-stationary array with scratchpads identical for every format (Balasubramonian et al., 2017), and we compare against a 32×32 FP16 reference at iso-area and at iso-throughput under our evaluation setting of 256×256 images and 50 sampling steps in Fig. 7.

Limitations. Our numbers come from logic synthesis rather than full physical design, so wiring is estimated and absolute area, delay and energy would shift in a commercial flow at a modern node. All formats share the same tools and constraints, and the estimate favors the baselines, whose decoders are harder to wire, so the comparison is more reliable than any single number. Our model also covers only the compute array, the quantizers and on-chip memory, and assumes the array is always busy. Under a memory-bound setting every 4-bit format moves the same bits per value and shares the same bottleneck, so the gains of PXP over full precision persist while the gaps among quantized formats narrow.

K Hardware Ablation

PXP's hardware efficiency stems from two design choices, an exponent-only decoder that stores one entry per exponent level, and a narrow MAC that exploits the explicit exponent-mantissa structure of each code. To isolate their contributions, Fig. A6 evaluates each in isolation against the full design, normalized to an FP8 (W8A8) accelerator.

The dominant cost shifts with activation bit-width. At W4A4, the MAC dominates. The full table holds only 16 entries and adds little overhead, so pairing it with the narrow MAC already yields $0.42\times$ FP8 latency, whereas an FP16 MAC makes PXP $2.6\times$ slower than FP8 despite the compact exponent decoder. At W4A8, the decoder dominates. The table grows to 256 entries, raising latency, energy and area to $55.8\times$, $35.9\times$ and $8.6\times$ FP8, while the wider mantissa brings the narrow MAC closer to FP16 and shrinks its advantage. At W4A6, the two overheads are comparable ($3.5\times$ vs. $4.4\times$ FP8 latency), so neither can be dropped. Combining both removes each bottleneck, as the

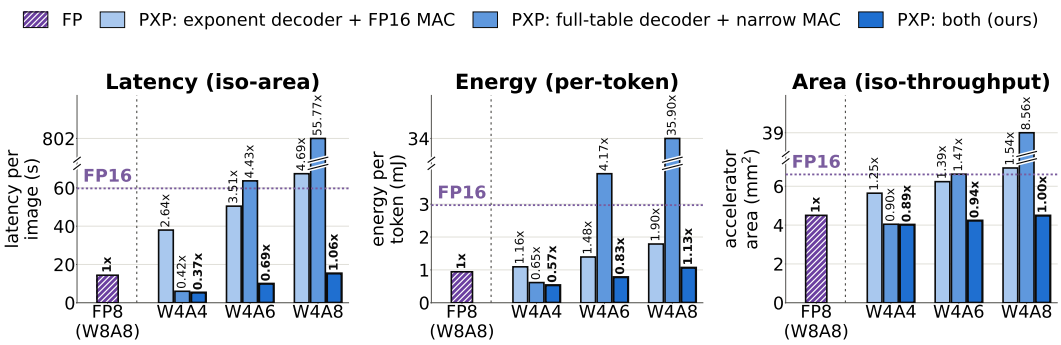


Figure A6: Hardware ablation of PXP on DiT-XL/2 (50 denoising steps). We separate the two sources of PXP's efficiency by pairing the exponent-only decoder with a standard FP16 MAC, and a full-table decoder with the narrow PXP MAC. Values are normalized to an FP8 (W8A8) accelerator, and the dotted line marks FP16. Broken axes indicate bars exceeding the plotted range.

exponent-only decoder keeps table cost flat with bit-width and the narrow MAC keeps arithmetic cheap. The full design is thus $2.7\times$ faster and $1.8\times$ more energy efficient than FP8 at W4A4, remains ahead at W4A6, and stays within 6% of FP8 latency at W4A8, with no area increase at any bit-width.

K.1 Model Size and Codebook Overhead

Fig. A5 reports model size and codebook storage for DiT-XL/2. Quantizing weights to 4 bits reduces the model from 1288 MiB in FP16 to 337 MiB for every format, since all formats store 4-bit weight codes. We therefore hold weights fixed and sweep activation precision, which isolates the storage that adaptivity adds.

Hardwired formats such as NF, AF and SF store a single codebook shared across the network, so their overhead is negligible by construction. PXP instead stores one prefix table per layer and denoising step, which multiplies the number of tables by the product of layers and steps. Each table, however, holds only one entry per exponent level rather than one per codeword, and the number of levels is bounded by $K_{\max} = 2^{b-1} - 1$. The total overhead therefore grows linearly with network depth and step count but remains small in absolute terms. As Fig. A5 shows, PXP stores three to four orders of magnitude more table data than a hardwired format, yet this stays below 1% of the quantized model at every activation bit-width.

K.2 Qualitative Results

Fig. A7 shows eight class-conditional samples from DiT-XL/2 on ImageNet at W4A6 with group size 128, using CFG = 1.5 and $T = 100$. PXP is visually closest to FP16, preserving object structure and texture, whereas the competing formats either blur the image or introduce severe distortion that alters the depicted object.

Fig. A8 shows eight text-to-image samples from PixArt-Σ on COCO prompts at W4A6 with per-token and per-channel quantization and $T = 20$. The same ordering holds. Most formats blur or fully distort the scene, and AF is the only competitive baseline, yet it loses fine detail and texture that PXP retains, keeping its samples close to FP16.

Fig. A9 shows five frames of a video generated by Open-Sora v1.2 on VBench at W4A4 with group size 128 and 50 sampling steps. At this precision most formats collapse entirely, producing flicker and incoherent frames. AF remains recognizable but heavily distorted, while PXP stays close to FP16 with only negligible artifacts and preserves both the content and the temporal consistency of the sequence.

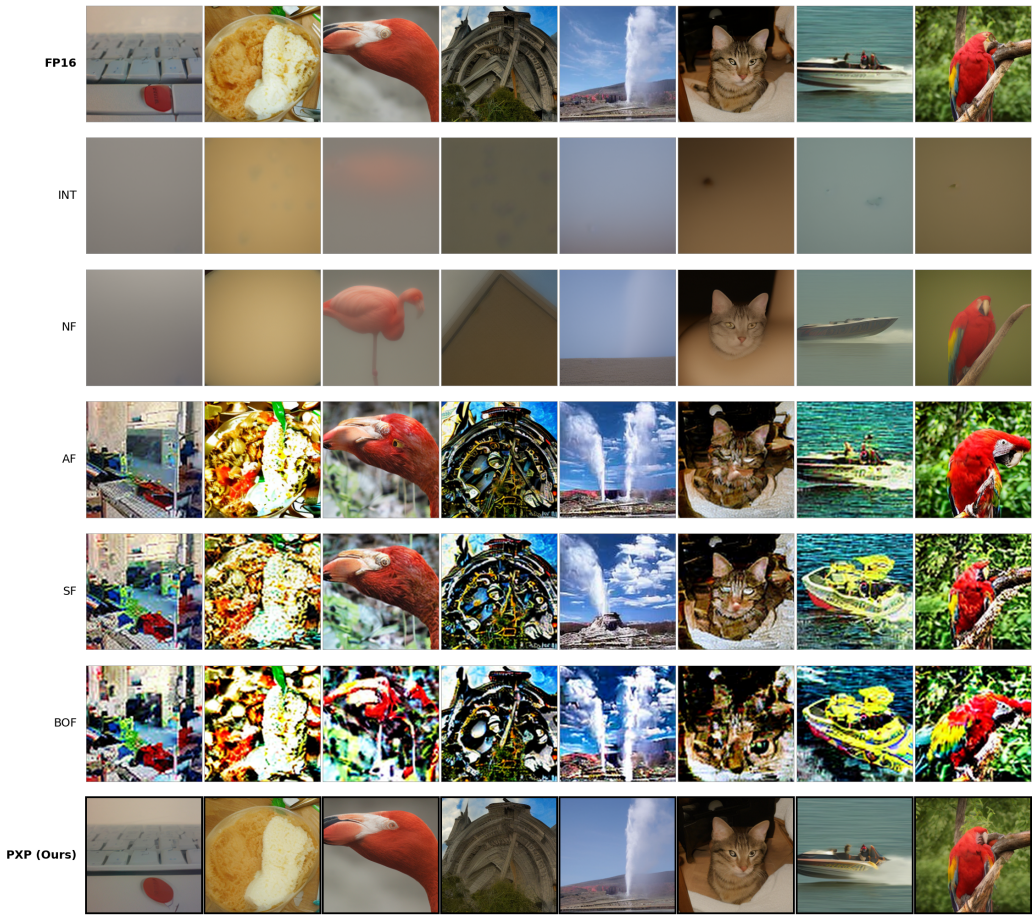


Figure A7: Qualitative comparison across quantization formats for DiT-XL/2 on ImageNet 256×256 at W4A6 with group size 128 (CFG 1.5, T=100).



Figure A8: Qualitative comparison across quantization formats for Pixart-$\sum$ on COCO at W4A6 (per-channel weights, per-token activations).



Figure A9: Qualitative comparison across quantization formats for OPEN-SORA V1.2 on VBench at W4A4 with group size 128.