GRAFT: HYPERDIMENSIONAL COMPUTING-BASED GRAPH RECONSTRUCTION VIA ALGEBRAIC FIXED TRANSFORMATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many scientific domains, chemistry foremost among them, represent their primary objects as attributed graphs, and learning on these objects typically requires mapping them to and from a fixed-dimensional continuous space. One important capability such representations support is graph reconstruction — encoding an attributed graph into a single vector and recovering it exactly — which underpins retrieval and structure generation. Existing solutions are learned encoder–decoder models, with graph autoencoders as the canonical example: training takes hundreds of GPU-hours and yields codecs that are confined to the distribution they were trained on. We introduce GRAFT (Graph Reconstruction via Algebraic Fixed Transformations), a graph encoder–decoder architecture that maps graphs to fixed-dimensional hypervectors and back through deterministic binding, bundling, and message-passing operations over a fixed set of random reference vectors: *the encoder and decoder contain no trainable parameters*. On a standard molecular-graph reconstruction benchmark, GRAFT recovers 93% of molecular graphs with up to 32 heavy atoms exactly and 91% of graphs with up to 64 heavy atoms, substantially improving on state-of-the-art learned autoencoders without any encoder or decoder training. Paired with standard task-specific learned heads, GRAFT is suitable for MoleculeNet property prediction, supports molecular generation on ZINC250K, and substantially outperforms baselines on the Fingerprint2Graph translation task.

1 INTRODUCTION

Autoencoders (AEs) play an essential role in modern representation learning: a high-fidelity encoder–decoder pair is trained once and then reused across many downstream tasks. In image generation, for example, many high-resolution systems first train an AE to compress raw pixels into a compact continuous space, and then train a separate generator to operate in that space (van den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021; Rombach et al., 2022). The same continuous representation also supports property-guided optimization (Nguyen et al., 2017; Engel et al., 2018), interpolation (White, 2016; Berthelot et al., 2019), and attribute editing (Shen et al., 2020; Karras et al., 2019).

Translating this approach to graphs is considerably harder: graphs vary in size, are invariant to node ordering, and carry discrete node and edge attributes that must be recovered exactly. A line of graph-level AEs from GraphVAE (Simonovsky & Komodakis, 2018) through PIGVAE (Winter et al., 2021) to the current state of the art, GRALE (Krzakala et al., 2025), has progressively improved reconstruction fidelity, but at the cost of hundreds of GPU-hours of training time and encoder–decoders tied to the specific data distribution they were trained on. The SOTA method GRALE even faces a hard technical limitation: its training loss scales as $\mathcal{O}(N^3)$ in the graph size N, which the authors identify as the reason their experiments cap at $N = 32$ nodes (Krzakala et al., 2025).

Hyperdimensional computing (HDC) (Kanerva, 2009; Kleyko et al., 2022), and in particular holographic reduced representations (HRR) (Plate, 1995), offers a parameter-free alternative: discrete objects are represented as high-dimensional random vectors and manipulated by three fixed operations — *binding* to compose, *bundling* to aggregate, and similarity search to query. While HDC has been productive outside graph learning, in edge-hardware classification (Rahimi et al., 2016b), biomedical signal processing (Rahimi et al., 2016a; Moin et al., 2021), and symbolic reasoning (Kanerva, 2010; Hersche et al., 2023b), applied to graphs it has so far been used only for encoding, as a

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.

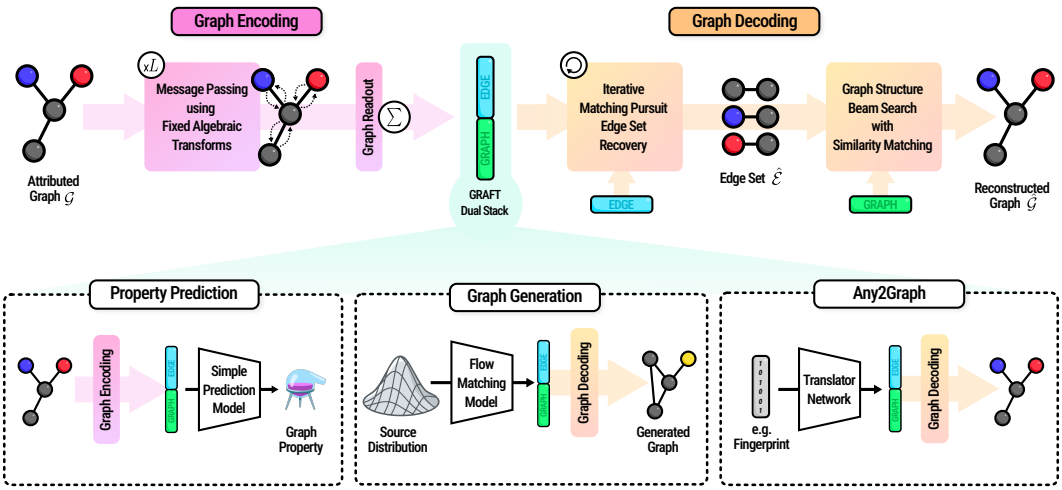


Figure 1: GRAFT method overview. **Top**: the parameter-free codec encodes a graph $\mathcal{G}$ to $\mathbf{h} = [\mathbf{h}_{\text{edge}} \| \mathbf{h}_{\text{graph}}]$ via fixed-transform message passing, and decodes $\hat{\mathcal{G}}$ by matching-pursuit edge recovery against $\mathbf{h}_{\text{edge}}$ followed by similarity-ranked beam search against $\mathbf{h}_{\text{graph}}$. **Bottom**: downstream tasks (property prediction, generation, Fingerprint2Graph) attach a learned component (prediction, flow matching, translator model) to the frozen codec output.

feature extractor for graph classification (Nunes et al., 2022; Kang et al., 2022; Dalvi & Honavar, 2025) and molecular property prediction (Teufel et al., 2026; Vergés et al., 2024; Jones et al., 2024). However, HDC representations offer the additional property of approximate invertibility: random high-dimensional vectors are near-orthogonal in expectation, so bundled components can be decoded by similarity search against the codebook (i.e., a lookup table that assigns a random hypervector to each atomic symbol), and binding admits an approximate algebraic inverse (*unbinding*) that undoes composition. These invertibility properties underpin resonator networks, for example, which iteratively factor products of hypervectors (Frady et al., 2020; Renner et al., 2024), as well as algebraic decoders for dense superpositions of bound symbols (Hersche et al., 2023a).

We introduce GRAFT (Graph Reconstruction via Algebraic Fixed Transformations), a graph encoder–decoder architecture that maps graphs to a fixed-dimensional hypervector through a deterministic HDC encoder over a random codebook and recovers them through an algebraic decoding algorithm: *the encoder and decoder contain zero trainable parameters*, and the codebook itself is sampled once from a fixed distribution rather than fitted to any dataset. On the molecular-graph reconstruction benchmarks used by GRALE (Krzakala et al., 2025), GRAFT substantially improves on state-of-the-art learned autoencoders in exact graph-isomorphism accuracy.

Throughout, we use GRAFT to refer specifically to the training-free graph codec: a fixed algebraic encoder that maps attributed graphs to hypervectors, paired with a fixed algebraic decoder that recovers the graph from the hypervector. At inference, the codec consists of the algebraic pipeline together with a random codebook of one hypervector per distinct node feature tuple — a fixed set of vectors drawn once from $\mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I}_D)$. When we apply the codec to downstream tasks (property prediction, fingerprint-to-graph translation, generation), we attach standard task-specific learned heads to the frozen codec output, following the canonical HDC pipeline of fixed encoders paired with learned heads (Rahimi et al., 2016b; Vergés et al., 2024), which exist to demonstrate that the training-free GRAFT representation is useful beyond reconstruction tasks.

Concretely, we present the following contributions:

- *A training-free graph codec.* GRAFT is graph encoder–decoder architecture without the need for prior data-based training (Sect. 4).
- *State-of-the-art reconstruction.* GRAFT exactly reconstructs 93% of graphs with up to 32 nodes (vs. 66.8% for the best learned baseline) on the PubChem benchmarks, and extends to 91% on PubChem64, a larger regime not reached by prior learned methods.
- *A reusable component for downstream tasks.* With a per-task learned head attached to the frozen GRAFT output, the same codec outperforms learned baselines on the Fingerprint2Graph (Krzakala

et al., 2024) translation task, can be applied to MoleculeNet property prediction (Wu et al., 2018), and unconditional molecular generation on ZINC250K (Irwin et al., 2012) drug-like molecules by utilizing a continuous flow matching model.

2 Related Work

Graph-Level Autoencoders. Learned graph-level autoencoders — GraphVAE (Simonovsky & Komodakis, 2018), PIGVAE (Winter et al., 2021), and the current state-of-the-art GRALE (Krzakala et al., 2025) — are fully trained encoder–decoder pairs whose parameters are bound to a target graph distribution. GRALE reports strong reconstruction on the PubChem16/PubChem32 benchmarks we adopt here. Node-level autoencoders such as VGAE (Kipf & Welling, 2016) are not directly comparable, as they do not produce a single graph-level vector. A separate line of work decodes molecules through chemistry-specific building blocks rather than arbitrary attributed graphs: junction trees of ring/atom motifs in JT-VAE (Jin et al., 2018) and its hierarchical successor HierVAE (Jin et al., 2020), a learned motif vocabulary in MoLeR (Maziarz et al., 2022), and SMILES strings decoded character-by-character (Gómez-Bombarelli et al., 2018) or via grammar production rules (Kusner et al., 2017). These decoders bake a molecular prior into their output space and are not directly comparable to general-purpose graph encoder–decoders such as GRALE or GRAFT.

Hyperdimensional Graph Representations. HDC has been applied to graphs primarily as an encoder-only featurizer for graph and node classification (Nunes et al., 2022; Kang et al., 2022; Dalvi & Honavar, 2025), molecular property prediction (Vergés et al., 2024), and binding-affinity estimation (Jones et al., 2024). Most directly, Teufel et al. (Teufel et al., 2026) introduce an HRR-based message-passing encoder used as a fixed molecular fingerprint for property prediction and similarity search, but not for reconstruction or generation. GRAFT builds on Teufel *et al.*'s encoder backbone but (i) replaces their manually curated global descriptors (graph size, diameter) with generic node-level structural features (node degree and random-walk return probabilities), (ii) splits the readout into a dual-stack edge term and graph term to make algebraic decoding tractable, and (iii) adds the matching-pursuit and beam-search decoder of Sect. 4.2 that reconstructs the graph from its hypervector — turning the prior encoder-only featurizer into a full codec.

3 Preliminaries

Hyperdimensional computing (HDC), also known as vector symbolic architectures (VSA), represents structured information as high-dimensional real-valued vectors called *hypervectors* (Kanerva, 2009; Kleyko et al., 2022). Among several VSA families (Kleyko et al., 2022), we adopt holographic reduced representations (HRR) (Plate, 1995). Their operations act on real-valued vectors in $\mathbb{R}^D$ and are computed via the fast Fourier transform.

Operations. HRR defines two core operations on hypervectors sampled i.i.d. from $\mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I}_D)$.

Binding ($\circledast$) associates two hypervectors via circular convolution:

$$(\mathbf{x} \circledast \mathbf{y})_k = \sum_{j=0}^{D-1} x_j \, y_{(k-j) \bmod D} \,. \tag{1}$$

Equivalently, $\mathbf{x} \circledast \mathbf{y} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{y}))$ where $\mathcal{F}$ is the discrete Fourier transform and $\odot$ element-wise multiplication, yielding $O(D \log D)$ complexity. Binding is commutative and associative, and the result is approximately orthogonal to both operands.

Bundling ($\oplus$) superimposes hypervectors by element-wise addition: $\mathbf{s} = \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \cdots \oplus \mathbf{v}_m$. The bundle $\mathbf{s}$ is similar to each constituent, enabling recovery via cosine similarity $\cos(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y} / (\|\mathbf{x}\| \, \|\mathbf{y}\|)$. Binding distributes over bundling: $\mathbf{x} \circledast (\mathbf{y} \oplus \mathbf{z}) = (\mathbf{x} \circledast \mathbf{y}) \oplus (\mathbf{x} \circledast \mathbf{z})$.

Key Properties. Two properties of HRR underpin the method developed in Sect. 4. *(i) Quasi-orthogonality*: for independent $\mathbf{x}, \mathbf{y} \sim \mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I}_D)$, the cosine similarity concentrates as $\cos(\mathbf{x}, \mathbf{y}) = O(1/\sqrt{D})$ with high probability; bound vectors $\mathbf{x} \circledast \mathbf{y}$ are likewise near-orthogonal to their operands. As a consequence, querying a bundle $\mathbf{s} = \bigoplus_{k=1}^{K} \mathbf{x}_k$ by similarity to a codebook entry returns the matching constituent against an interference of $K - 1$ near-orthogonal terms, each contributing

variance $\sim 1/D$, for a total cross-talk standard deviation of $O(\sqrt{K/D})$ — the mechanism that lets matching pursuit (Eq. 5) identify the next edge. Once a constituent has been identified, subtracting it from the unnormalized bundle is algebraically exact and leaves the remaining terms intact. ***(ii) Capacity***: the number of components K that can be reliably retrieved from a bundle scales linearly with the dimension D and inversely with $\log M$, where M is the codebook size; concretely $K = O(D/\log M)$ for vanishing per-item retrieval error (Plate, 1995; Frady et al., 2018), linking hypervector dimension directly to representational capacity.

These primitives allow GRAFT to deterministically map attributed graphs to real-valued vectors and recover discrete structure from continuous representations, as described next.

4 METHOD

Here, we introduce GRAFT, a method that encodes and decodes graph structures to and from a fixed-size vector representation. The encoding integrates hyperdimensional binding and bundling operations into an iterative message-passing scheme. Graph structures are subsequently reconstructed from this fixed-size vector representation by leveraging algebraic properties of the encoding to recover the graph's edge set, and then assembling those edges into a graph through a similarity-informed heuristic beam search. The encoder and decoder described below contain no trainable parameters and are constructed without reference to any dataset.

4.1 GRAPH ENCODING

We build on the method introduced by Teufel et al. (2026) to derive a fixed-size graph representation $\mathbf{h} \in \mathbb{R}^{2D}$ from a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ with node set $\mathcal{V} = \{0, \dots, V\}$, edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and node features $\mathcal{X}$. The encoding can be summarized in three steps: node initialization, message passing, and graph readout.

Node Initialization. Each node $v_i \in \mathcal{V}$ is characterized by a discrete feature tuple $\mathcal{X}_i = (x_1^i, x_2^i, \dots, x_n^i)$ whose components mix domain-specific node attributes with generic graph-structural ones. For molecular graphs, we use the atom type, the total hydrogen count, the formal charge, and ring membership as domain-specific components, and the node degree together with the *random-walk return probabilities (RW)* (Li et al., 2020; Dwivedi et al., 2022; Ma et al., 2023) $P_{\mathrm{rw}}^{(k)}$ at even walk lengths $k \in \{2, 4, 6, \dots\}$, discretized into b bins, as generic components. The *node codebook* assigns one i.i.d. random hypervector $\sim \mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I}_D)$ to each distinct feature tuple, so that two nodes with identical tuples share the same initial hypervector. Node v_i's initial hypervector $\mathbf{x}_i^{(0)} \in \mathbb{R}^D$ is the draw corresponding to its tuple $\mathcal{X}_i$.

Message Passing. Starting from $\mathbf{x}_i^{(0)}$, we run L rounds of message passing. In round $l \in \{1, \dots, L\}$, each neighbor $j \in \mathcal{N}(i)$ contributes a message obtained by binding its hypervector to the center node's, the messages are aggregated by bundling, and the result is normalized before it becomes the round-l representation:

$$\mathbf{x}_i^{(l)} = \mathrm{norm}\left(\bigoplus_{j \in \mathcal{N}(i)} \left(\mathbf{x}_i^{(l-1)} \circledast \mathbf{x}_j^{(l-1)} \right) \right), \tag{2}$$

where $\mathcal{N}(i)$ denotes the set of neighbors of v_i and $\mathrm{norm}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$ is the ℓ_2 unit normalization. The per-round normalization is necessary because the expected magnitude of $\mathbf{x}_i^{(l)}$ grows with l from repeated binding, and without it the deepest layers would dominate any subsequent aggregation across depths. Bundling is permutation-invariant in its arguments, so the update does not depend on the order in which $\mathcal{N}(i)$ is enumerated, and the quasi-orthogonality of HRR keeps the individual center to neighbor bindings distinguishable inside the aggregate. Because each round extends the receptive field by one hop, $\mathbf{x}_i^{(L)}$ encodes the L-hop subgraph rooted at v_i, and the depth L therefore sets the structural scope of the encoder. In contrast to a message-passing neural network, the aggregation and update functions here are the fixed HDC primitives of Sect. 3 rather than learned modules.

Graph Readout. We collapse the per-node hypervectors into two complementary graph-level terms: an edge term $\mathbf{h}_{\text{edge}}$ and a graph term $\mathbf{h}_{\text{graph}}$. The edge term is obtained by bundling the round-1

representations over the node set; expanding eq. 2 gives

$$\mathbf{h}_{\text{edge}} = \bigoplus_{v_i \in \mathcal{V}} \mathbf{x}_i^{(1)} = \bigoplus_{v_i \in \mathcal{V}} \left(\bigoplus_{j \in \mathcal{N}(i)} \mathbf{x}_i^{(0)} \circledast \mathbf{x}_j^{(0)} \right), \tag{3}$$

which is a superposition of all edge-bindings in $\mathcal{G}$. The graph term, on the other hand, captures multi-scale structural information by aggregating per-round node bundles across all depths,

$$\mathbf{h}_{\text{graph}} = \bigoplus_{l=0}^{L} \bigoplus_{v_i \in \mathcal{V}} \mathbf{x}_i^{(l)}. \tag{4}$$

Because the round-l representations $\mathbf{x}_i^{(l)}$ are already normalized by Eq. 2, the graph term spans a superposition of node-type hypervectors at $l = 0$ through a full L-hop structural fingerprint at $l = L$, with balanced contribution from every message-passing depth.
Finally, we define the concatenation $\mathbf{h} = [\mathbf{h}_{\text{edge}} \| \mathbf{h}_{\text{graph}}] \in \mathbb{R}^{2D}$ as the GRAFT *dual stack* representation, which is subsequently used as the basis of the graph reconstruction procedure, property prediction and graph generation. $\mathbf{h}_{\text{edge}}$ carries the information needed to recover individual edges, while $\mathbf{h}_{\text{graph}}$ serves as a global fingerprint used to rank candidate reconstructions during decoding.

4.2 HDC Decoding

We reconstruct the original graph from $\mathbf{h}$ in two stages: edge recovery against $\mathbf{h}_{\text{edge}}$ produces a list of candidate edges, over which a beam search then ranks partial reconstructions against $\mathbf{h}_{\text{graph}}$.

Iterative Edge Recovery. By Eq. 3, $\mathbf{h}_{\text{edge}}$ is a superposition of pairwise bindings $\mathbf{x}_a \circledast \mathbf{x}_b$ of initial node hypervectors. We recover edges by iteratively matching the residual against these bindings and subtracting the matched component — an instance of *matching pursuit* (Mallat & Zhang, 1993) over the dictionary of pairwise edge bindings. Starting from $\mathbf{r} \leftarrow \mathbf{h}_{\text{edge}}$ and $\hat{\mathcal{E}} \leftarrow \emptyset$, each step picks the pair (a, b) whose binding is most aligned with the residual under cosine similarity,

$$(a^*, b^*) = \arg\max_{a,b} \cos\big(\mathbf{r},\ \mathbf{x}_a \circledast \mathbf{x}_b\big), \tag{5}$$

appends it to $\hat{\mathcal{E}}$, and subtracts the matched component from the residual: $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{x}_{a^*} \circledast \mathbf{x}_{b^*} - \mathbf{x}_{b^*} \circledast \mathbf{x}_{a^*}$, where the factor two reflects that each undirected edge contributes its endpoint binding twice to $\mathbf{h}_{\text{edge}}$ via Eq. 3. Cosine (rather than inner product) keeps successive iterations comparable as $\|\mathbf{r}\|$ shrinks, and quasi-orthogonality ensures that the subtraction leaves the remaining edge-bindings nearly intact, so each iteration exposes the next-strongest signal. The loop halts when the residual norm $\|\mathbf{r}\|_2$ either drops below a small threshold ε or fails to decrease relative to the previous iteration, returning $\hat{\mathcal{E}}$ as input to the subsequent beam search.

Beam Search Reconstruction. $\hat{\mathcal{E}}$ fixes the multiset of *edge types* but not a topology: typed edges may or may not share endpoints, and the node multiset depends on how they are wired. We resolve the assembly by beam search against the graph term $\mathbf{h}_{\text{graph}}$.
We initialize the beam with every edge of $\hat{\mathcal{E}}$ as its own single-edge seed, and grow candidate graphs one *node* at a time. At each iteration every partial graph G in the beam selects the node $a^\star \in V(G)$ with smallest residual degree (the gap between its target degree, read from its feature tuple, and its current degree in G) as the active anchor. For each leftover atom-type t such that the typed edge $(\text{type}(a^\star), t)$ is still available in $\hat{\mathcal{E}}_{\text{rem}}$, we attach a new node of type t to $a^\star$, and additionally enumerate every non-empty subset of the remaining anchors as candidate cycle-closing back-edges from the new node, subject to type and residual-degree availability. Each extension G' is scored against the target fingerprint as $s(G') = \cos(\mathbf{h}_{\text{graph}}, \mathbf{h}_{G'})$, where $\mathbf{h}_{G'}$ is the graph term obtained by re-encoding G' through the encoder of Sect. 4.1. We retain the top-B_{beam} partial graphs per edge-count group; once every node has been placed, the highest-scoring complete graph is returned. For molecular graphs, bond orders are inferred post-hoc from node attributes via chemical valency rules. A more detailed explanation of the decoding algorithm can be found in the supplementary materials.

5 Experiments

5.1 Graph Reconstruction

Setup. We follow GRALE's reconstruction protocol (Krzakala et al., 2025) on PubChem16 (14M graphs, ≤ 16 heavy atoms) and PubChem32 (84M graphs, ≤ 32 heavy atoms), and additionally introduce PubChem64 (118M graphs, ≤ 64 heavy atoms) as an extended regime. As a metric we graph isomorphism accuracy (GI Acc.), which determines how often the exact same graph is reconstructed. The baselines are GraphVAE (Simonovsky & Komodakis, 2018), PIGVAE (Winter et al., 2021), and GRALE (Krzakala et al., 2025), with numbers taken from the GRALE paper. GRAFT uses a single *headline configuration* across all three benchmarks ($D = 1024$, $L = 5$, $\mathcal{K} = \{4, 8, 12, 18\}$ with $B = 8$ bins, $B_{\text{beam}} = 32$; full settings in the supplementary material).

Table 1: Graph reconstruction on PubChem (GI Acc., %, mean±std over three seeds, n=10 000 test molecules per seed). Baselines from (Krzakala et al., 2025). Best per column in **bold**. * PIGVAE uses ground-truth graph size. † GraphVAE too expensive to train at PubChem32 scale.

Model	PubChem16	PubChem32	PubChem64
GraphVAE (2018)	7.80	N.A.†	–
PIGVAE* (2021)	41.00	24.91	–
GRALE (2025)	93.00	66.80	–
GRAFT	$\mathbf{98.5}_{\pm 0.32}$	$\mathbf{92.9}_{\pm 0.44}$	$\mathbf{90.7}_{\pm 0.12}$

Discussion. Across all three benchmarks, GRAFT substantially improves on state-of-the-art learned graph autoencoders (see Table 1). The smaller PubChem16 benchmark shows near-perfect reconstruction fidelity at 98.52% GI Acc. The gap is most apparent on PubChem32, where GRAFT reaches 92.87% against GRALE's 66.80% — a 26.1 percentage point absolute improvement over the strongest baseline. On PubChem64, GRAFT retains 90.74% GI Acc. in a regime where no learned baseline has been evaluated yet, due to training-time scaling bottleneck (Sect. 1); GRAFT, having no training step, only needs the decoder to handle a larger candidate space at inference time. The trade-off appears at inference: GRAFT's decoder runs iterative matching pursuit followed by beam search, placing per-molecule decoding time roughly an order of magnitude above GRALE's single-shot transformer decoder and scaling further with graph size and beam budget.

Despite the high accuracy at the dataset level, the per-graph GI Acc. drops sharply for graphs with 40+ heavy atoms (Figure 2b). In this large-graph regime the dominant cause is decoder search rather than encoder capacity: even across PubChem64, the correct edge set $\hat{\mathcal{E}}$ is retrieved with a near-perfect accuracy of 99.9999%, but as graph size and the number of possible edge combinations increase, the beam search is less likely to find the correct configuration with a limited beam budget. Decoding is additionally subject to a theoretical 1-WL ceiling: with L rounds of message passing, the encoder is at most as expressive as L iterations of the 1-dimensional Weisfeiler–Leman test (Xu et al., 2019; Morris et al., 2019), so non-isomorphic graphs that 1-WL cannot distinguish in L iterations are mapped to indistinguishable hypervectors. A detailed analysis of the decoding failure modes can be found in the Supplementary Materials.

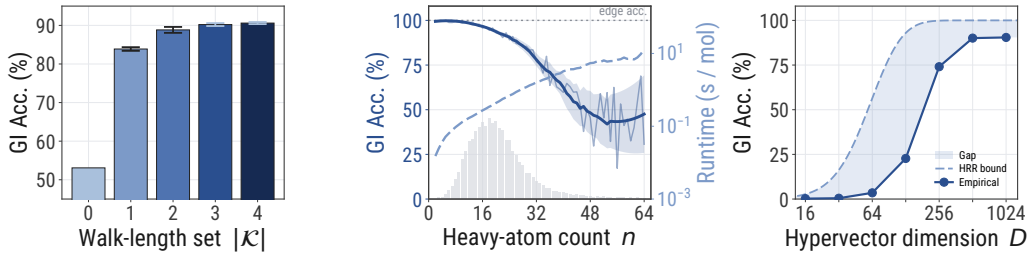


Figure 2: Ablation. (a) GI Acc. vs. the number of random walk steps used to construct the node structural descriptor on PubChem32; $|\mathcal{K}| = 0$ corresponds to disabling random-walk return probabilities entirely. (b) GI Acc. vs. heavy-atom count n on PubChem64; the gray histogram shows the dataset's size distribution and the dashed gray line marks the average per-edge accuracy. (c) GI Acc. vs. hypervector dimension D on PubChem32; the shaded region marks the gap between the empirical curve and the HRR capacity bound.

5.2 FINGERPRINT2GRAPH

Setup. We evaluate on the Fingerprint2Graph task of Any2Graph (Krzakala et al., 2024): predict a molecular graph from its ECFP4 fingerprint (Rogers & Hahn, 2010), which hashes a molecule's atom-centered substructures up to radius 2 into a fixed-length bit vector. This local descriptor retains which substructures appear but discards their connectivity. We evaluate on QM9 (Ramakrishnan et al., 2014) ($\sim$134K molecules, $\leq$ 9 heavy atoms) and the >1.3M-molecule `ABCDEFGH` subset (Blum et al., 2011) of GDB13 (Blum & Reymond, 2009) ($\leq$ 13 heavy atoms).

Table 2: Fingerprint2Graph (GI Acc., %, ↑). * uses ground-truth graph size. "+FD": Feature Diffusion variant. Baselines from (Krzakala et al., 2024; 2025).

Model	QM9	GDB13
Trained from scratch		
FGWBary-NN* (2022)	1.00	N.A.
FGWBary-ILE* (2022)	7.10	N.A.
FGWBary-ILE* + FD (2022)	28.95	N.A.
Relationformer (2022)	0.05	0.00
Relationformer + FD (2022)	9.95	0.01
Any2Graph (2024)	7.50	0.05
Any2Graph + FD (2024)	29.85	16.25
Pretrained on PubChem32		
GRALE (2025)	30.77	32.02
GRALE + Finetuning (2025)	35.27	53.22
GRAFT	**83.90**	**71.20**

GRAFT Adaptation. Fingerprint2Graph requires translating an external lossy representation into a graph; the codec alone cannot do this because the input does not lie in the codec's domain. We therefore attach a single learned MLP projector $\psi\colon \mathbb{R}^{d_{\text{fp}}} \to \mathbb{R}^{2D}$ that maps an input fingerprint to the HDC encoding of its ground-truth graph. At inference we project a fingerprint through ψ and decode the resulting hypervector through the frozen GRAFT decoder of Sect. 4.2. ψ is the only trainable component; the GRAFT encoder and decoder remain training-free and frozen.

Discussion. GRAFT substantially outperforms all Fingerprint2Graph baselines on both datasets, despite adding only a projector MLP on top of the frozen HDC pipeline (see Table 2), while GRALE's previous SOTA results required finetuning of the underlying encoder layers. On QM9 GRAFT reaches 83.90% GI Acc., a 48.6-point absolute improvement over the strongest learned baseline. A similar gap holds on GDB13 (71.20%, a 18.0-point margin).
Since Sect. 5.1 shows the HDC decoder to be near-perfect at the size scale covered by both datasets (≤ 16 nodes), we hypothesize that the residual error on Fingerprint2Graph is driven primarily by how closely ψ can recover the target hypervector from the lossy ECFP4 input, rather than by any decoding-stage limitation. The two dominant error sources are likely the finite training data which the projection function has to inferred from and the inherent information loss in the fingerprint itself. Hash collisions and the radius-2 truncation cause some non-isomorphic molecules to share identical inputs, bounding any fingerprint-based method from above regardless of the decoder used downstream.

5.3 PROPERTY PREDICTION

Setup. We evaluate the frozen GRAFT hypervectors as a graph-level representation on MoleculeNet (Wu et al., 2018), training a single MLP head per task on top of the frozen codec — regression for QM9, QM40 (Madushanka et al., 2024), ESOL, LIPO, and FreeSolv, and classification for BBBP and BACE.

Discussion. Prior work has already shown that HDC encoders can serve as effective molecular descriptors for property prediction (Teufel et al., 2026; Vergés et al., 2024); the question here is whether the reconstruction-oriented codec output of GRAFT carries the same downstream utility. Without the random-walk descriptor, frozen GRAFT hypervectors perform comparably to the learned graph-AE baselines: GRAFT (w/o RW) is competitive on the classification (BBBP, BACE) and regression tasks (QM9, QM40, LIPO).
Including the random-walk descriptor, however, leads to a substantial degradation on the regression tasks (e.g. ESOL $0.289 \to 0.472$, LIPO $0.451 \to 0.563$, FreeSolv $0.310 \to 0.574$), placing GRAFT below the learned baselines across these columns. This stands in tension with the reconstruction result of Sect. 5.1, where the RW descriptor is critical for the encoder to reach the reported GI Acc. (Fig. 2a): the same feature that benefits reconstruction substantially degrades downstream property prediction. We hypothesize that the RW features improve reconstruction accuracy because they

Table 3: MoleculeNet property prediction using frozen graph-level representations pretrained on PubChem32. Regression metric is normalized MAE (lower is better, ↓); classification metric is ROC-AUC (higher is better, ↑). Values are means over 5 train/test splits; standard deviations are omitted here for space and reported in the supplementary material. Baseline numbers are taken directly from (Krzakala et al., 2025) (Table 3). Best per column in **bold**.

	Regr. (norm. MAE ↓)					Class. (ROC-AUC ↑)	
Model	QM9 (U_0)	QM40 (U_0)	ESOL	LIPO	FreeSolv	BBBP	BACE
SimGRACE (2022)	0.110	0.025	0.293	0.534	0.374	0.745	0.866
InfoGraph (2020)	0.122	0.020	**0.255**	0.495	0.297	0.729	0.845
GraphMAE (2022)	0.222	0.247	0.291	0.527	0.378	0.773	0.857
GVAE (2016)	0.765	0.328	0.306	0.668	0.344	0.705	0.771
PIGVAE (2021)	0.031	0.019	0.279	0.523	0.283	0.675	0.816
GRALE (2025)	0.015	0.018	0.274	0.511	**0.272**	0.731	0.821
GRAFT	0.119	0.196	0.472	0.563	0.574	0.848	0.846
GRAFT (w/o RW)	**0.011**	**0.0075**	0.289	**0.451**	0.310	**0.857**	**0.860**

substantially increase the number of distinct node-feature combinations, making individual nodes and edges within a single molecule appear more "unique" and thereby shrinking the search space for the beam search. The same mechanism likely prevents a property-prediction model from effectively generalizing beyond the training set.
Resolving this tension between reconstruction capability and downstream prediction performance is left to future work on the core methodology.

5.4 Unconditional Generation

Setup. We further evaluate the possibility of using GRAFT's embedding space as the source of molecular generation on the ZINC250K dataset (∼250K drug-like molecules, average 23.2 atoms) (Irwin et al., 2012).
In the base setting, we train a generative flow-matching model (Lipman et al., 2023) on the empirical distribution of GRAFT's hypervectors for ZINC250K molecules and decode drawn samples into graph structures using the decoding pipeline of Sect. 4.2. Inspired by the latent-diffusion / VQ-GAN paradigm of image generation (Rombach et al., 2022; Esser et al., 2021; van den Oord et al., 2017), in addition to the base setting, we also train a simple Autoencoder on the ZINC250K hypervectors and use a flow matching model to subsequently sample in the AE's latent space.
We evaluate using standard molecular-generation metrics: validity, uniqueness, novelty, their joint NUV score, internal diversity (ID1/ID2), and the distributional-alignment distances ΔQED, ΔSAS, measuring how far the sampled-molecule means for drug-likeness (QED) and synthetic accessibility (SAS) drift from the ZINC250K mean. Baselines span VAE, graph-VAE, autoregressive-transformer, and diffusion paradigms.

Discussion. GRAFT achieves essentially near-perfect validity, uniqueness, novelty, and internal diversity on ZINC250K (Table 4) in both the base and AE settings, demonstrating the general feasibility of graph generation through fixed-size HDC vector representations. The base setting, however, exhibits a significant drift from ZINC's drug-like distribution ($\Delta\text{SAS} = 1.77$). In practice this is reflected by generated molecules whose atom-type frequencies match the dataset but which show an increased occurrence of non-standard ring configurations. The AE setting mitigates this drift somewhat, achieving distributional alignment comparable to previous generative methods from the literature ($\Delta\text{SAS} = 0.64$). We hypothesize that this difference is a consequence of the HDC-derived embedding space: because the binding operations produce quasi-orthogonal vectors, the resulting space is strongly distributed over all dimensions and quasi-discrete rather than a well-defined manifold, making it a particularly difficult target for a flow-matching model with an isotropic Gaussian source distribution. The compression provided by the autoencoder, by contrast, induces a smoother latent geometry that is likely more accessible to the flow, leading to an improvement despite the autoencoder adding an additional noise source to the reconstruction pipeline.

Table 4: Unconditional generation on ZINC250K. ΔQED and ΔSAS are absolute distances between the generated-sample mean and the ZINC250K dataset mean ($\mu_{\text{QED}} = 0.73$, $\mu_{\text{SAS}} = 3.05$). Best per column in **bold**. ‡ Val./Uniq./Nov./ID1/ID2/QED/SAS from MG-Diff (Zhang et al., 2025) (Table 1); NUV, ΔQED, ΔSAS computed from those.

Model	Val. ↑	Uniq. ↑	Nov. ↑	NUV ↑	ID1 ↑	ID2 ↑	ΔQED ↓	ΔSAS ↓
JT-VAE (2018)‡	**100.0**	**100.0**	89.3	89.3	0.863	0.852	0.01	**0.05**
RGCVAE (2025)	99.9	61.1	**100.0**	61.0	**0.989**	–	0.30	–
MolGPT (2022)‡	97.3	98.1	86.4	82.5	0.851	0.843	0.03	0.42
DiGress (2023)‡	85.2	**100.0**	99.9	85.1	0.855	0.849	**0.00**	0.33
MG-Diff (2025)	96.4	99.4	98.5	94.4	0.891	0.881	0.03	0.78
GRAFT +FM	**100.0**	99.9	99.9	**99.9**	0.925	**0.938**	0.14	1.77
GRAFT +AE +FM	99.5	99.7	99.6	99.1	0.918	0.933	0.17	0.62

6 LIMITATIONS

Decoder scalability. The algebraic decoder of Sect. 4.2 is linear in the number of codebook entries per edge and combinatorial in topology reconstruction. As graphs grow, the candidate space that pattern matching and beam search must explore grows with it, and we see this reflected empirically as a sharp accuracy drop-off above 40 heavy atoms in the PubChem64 regime (Sect. 5.1). Two avenues appear promising for scaling GRAFT beyond 64 nodes. The first is to shrink the candidate space at its source: augmenting the node features with random-walk descriptors already lifts reconstruction accuracy substantially in our current pipeline (Fig. 2a), at the cost of a larger codebook and reduced downstream property-prediction generalization (Sect. 5.3); other feature combinations along these lines are worth exploring. The second is to handle the remaining search more efficiently. An option here would be to introduce a general purpose learned component to more efficiently suggest candidate proposals or pruning options during the search.

1-WL expressiveness ceiling. The encoder is built from L rounds of message passing and is therefore at most as expressive as L iterations of the 1-Weisfeiler–Leman test (1-WL); non-isomorphic graphs that 1-WL cannot distinguish are mapped to indistinguishable hypervectors and constitute a theoretical ceiling on reconstruction accuracy. In practice this ceiling still permits approximately 93% exact reconstruction on PubChem32 (Sect. 5.1). Although the introduction of the random walk features already addresses this problem to a certain degree, a breakdown of the failure modes attributes roughly 74% of the residual reconstruction errors to encoder isomorphisms of this kind, with beam-search failures overtaking the encoder-isomorphism contribution at the upper end of the PubChem64 size range. Lifting this limitation will most likely require an encoder to capture global graph structure beyond what is achievable with a local message-passing paradigm.

7 CONCLUSION

In summary we have presented GRAFT, a training-free graph encoder–decoder architecture built from fixed algebraic operations based on hyperdimensional computing, achieving state-of-the-art results in graph reconstruction and Fingerprint2Graph mapping, while also being applicable to property prediction and graph generation.

When GRAFT is used as the basis for molecular generation, a key advantage is that graph size and topology are jointly encoded in a single fixed-size vector and can therefore vary together during sampling. This is particularly useful for property-conditioned generation with properties coupled to graph size (e.g. solubility). While GRAFT-based molecular generation does not yet match the quality of state-of-the-art discrete diffusion and flow-matching models, we expect the gap to close through architectural variations of the generative model and engineering refinements of the GRAFT search algorithm which the decoder is based on.

The strong Fingerprint2Graph result furthermore suggests potential applications beyond reconstruction: If any fixed-size representation can be mapped into the codec's latent through a single MLP, this may enable cross-latent-space graph optimization within the latent spaces of existing pre-trained GNN models without the need for retraining.

AI USE STATEMENT

In this work, we used generative AI tools (Claude, including Claude Code; Anthropic) for the following tasks with required disclosure: developing the conceptual framework of the method, formulating mathematical claims, designing and providing feedback on the experimental methodology, implementing methods, cleaning and reformatting datasets, supporting qualitative analysis of the results, and interpreting results. We have not used generative AI tools to provide critical ingredients for proving mathematical claims, to assist in the writing of proofs, or to propose or refine hypotheses; generating synthetic datasets and translation are not applicable to this work.

Additionally, we used generative AI tools for the following tasks with recommended disclosure: drafting parts of the paper, editing the paper to improve readability, suggesting the structure and title of the paper, formatting references, identifying and summarizing relevant literature, sourcing and searching for information, suggesting experimental parameters, creating and editing software code, creating and modifying scientific figures, and creating auxiliary artifacts such as the manuscript build tooling.

We have reviewed all AI-assisted work. AI-assisted code was covered by unit and integration tests and validated against reference results; all conceptual and methodological decisions, including the final framing of the method and the interpretation of the experimental results, were made by the authors; and all references suggested with AI assistance were verified against the original sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or newly collected data. All experiments use publicly available molecular-graph benchmarks (PubChem, QM9, GDB13, MoleculeNet, ZINC250K), which we use under their published terms and cite in the paper. GRAFT is a general-purpose graph encoder–decoder; its evaluation domain is molecular structure, and we are not aware of a direct pathway from exact graph reconstruction to harmful applications beyond those that already exist for the underlying public datasets. The authors declare no conflicts of interest or sponsorship that could bias the presented results.

REPRODUCIBILITY STATEMENT

The codec has no trainable parameters, so every result depends only on the fixed configuration and the random seed. The datasets and their preprocessing are described in Sect. D.1, the exact codec configuration used for all reconstruction results in Sect. D.2, and the codebook construction, iterative edge recovery, and beam-search decoding in Sects. D.3–D.5. The learned task heads for Fingerprint2Graph and property prediction are specified in Sect. D.7 and the generation pipeline in Sect. D.8. Compute requirements and the software environment, including code availability, are given in Sects. D.9 and D.10. All ablations report the single-axis sweeps around a stated center configuration (Sect. E).

REFERENCES

Viraj Bagal, Rishabh Aggarwal, P. K. Vinod, and U. Deva Priyakumar. MolGPT: Molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling*, 62(9):2064–2076, 2022. doi: 10.1021/acs.jcim.1c00600.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1807.07543.

Lorenz C. Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, 131(25): 8732–8733, 2009. doi: 10.1021/ja902302h.

Lorenz C. Blum, Ruud van Deursen, and Jean-Louis Reymond. Visualisation and subsets of the chemical universe database GDB-13 for virtual screening. *Journal of Computer-Aided Molecular Design*, 25(7):637–647, 2011. doi: 10.1007/s10822-011-9436-y.

Luc Brogat-Motte, Rémi Flamary, Céline Brouard, Juho Rousu, and Florence d'Alché Buc. Learning to predict graphs with fused Gromov-Wasserstein barycenters. In *International Conference on Machine Learning (ICML)*, pp. 2321–2335, 2022.

Guojing Cong, Tom Potok, Hamed Poursiami, and Maryam Parsa. HyperGraphX: Graph transductive learning with hyperdimensional computing and message passing. *arXiv preprint arXiv:2510.23980*, 2025.

Abhishek Dalvi and Vasant Honavar. Hyperdimensional representation learning for node classification and link prediction. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 88–97. ACM, 2025. doi: 10.1145/3701551.3703492.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations (ICLR)*, 2022.

Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. In *International Conference on Learning Representations (ICLR)*, 2018. arXiv:1711.05772.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. arXiv:2012.09841.

E. Paxon Frady, Denis Kleyko, and Friedrich T. Sommer. A theory of sequence indexing and working memory in recurrent neural networks. *Neural Computation*, 30(6):1449–1513, 2018. doi: 10.1162/neco_a_01084.

E. Paxon Frady, Spencer J. Kent, Bruno A. Olshausen, and Friedrich T. Sommer. Resonator networks, 1: An efficient solution for factoring high-dimensional, distributed representations of data structures. *Neural Computation*, 32(12):2311–2331, 2020.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018. doi: 10.1021/acscentsci.7b00572.

Michael Hersche, Zuzanna Opala, Geethan Karunaratne, Abu Sebastian, and Abbas Rahimi. Decoding superpositions of bound symbols represented by distributed representations. In *17th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, 2023a. CEUR-WS Vol-3432.

Michael Hersche, Mustafa Zeqiri, Luca Benini, Abu Sebastian, and Abbas Rahimi. A neuro-vector-symbolic architecture for solving Raven's progressive matrices. *Nature Machine Intelligence*, 5: 363–375, 2023b.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021. arXiv:2207.12598.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52 (7):1757–1768, 2012. doi: 10.1021/ci3001277.

Jan H. Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical Science*, 10:3567–3572, 2019. doi: 10.1039/C8SC05372C.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *PMLR*, pp. 2323–2332, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *PMLR*, pp. 4839–4848, 2020.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *PMLR*, pp. 10362–10383, 2022.

Derek Jones, Xiaohua Zhang, Brian J. Bennion, Sumukh Pinge, Weihong Xu, Jaeyoung Kang, Behnam Khaleghi, Niema Moshiri, Jonathan E. Allen, and Tajana S. Rosing. HDBind: Encoding of molecular structure with hyperdimensional binary representations. *Scientific Reports*, 14, 2024. doi: 10.1038/s41598-024-80009-w.

Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009. doi: 10.1007/s12559-009-9009-8.

Pentti Kanerva. What we mean when we say "what's the dollar of Mexico?": Prototypes and mapping in concept space. In *AAAI Fall Symposium: Quantum Informatics for Cognitive, Social, and Semantic Processes*, 2010.

Jaeyoung Kang, Minxuan Zhou, Abhinav Bhansali, Weihong Xu, Anthony Thomas, and Tajana Rosing. RelHD: A graph-based learning on FeFET with hyperdimensional computing. In *IEEE 40th International Conference on Computer Design (ICCD)*, pp. 553–560. IEEE, 2022. doi: 10.1109/ICCD56317.2022.00087.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. arXiv:1812.04948.

Spencer J. Kent, E. Paxon Frady, Friedrich T. Sommer, and Bruno A. Olshausen. Resonator networks, 2: Factorization performance and capacity compared to optimization-based methods. *Neural Computation*, 32(12):2332–2388, 2020.

Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, et al. PubChem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, 2016. doi: 10.1093/nar/gkv951.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016. arXiv:1611.07308.

Denis Kleyko, Dmitri A. Rachkovskij, Evgeny Osipov, and Abbas Rahimi. A survey on hyperdimensional computing aka vector symbolic architectures, part I: Models and data transformations. *ACM Computing Surveys*, 55(6):1–40, 2022. doi: 10.1145/3538531.

Paul Krzakala, Junjie Yang, Rémi Flamary, Florence d'Alché Buc, Charlotte Laclau, and Matthieu Labeau. Any2Graph: Deep end-to-end supervised graph prediction with an optimal transport loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2402.12269.

Paul Krzakala, Gabriel Melo, Charlotte Laclau, Florence d'Alché Buc, and Rémi Flamary. The quest for the GRAph level autoEncoder (GRALE). In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.22109.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *PMLR*, pp. 1945–1954, 2017.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

László Lovász. Random walks on graphs: A survey. In D. Miklós, V. T. Sós, and T. Szőnyi (eds.), *Combinatorics, Paul Erdős is Eighty*, volume 2, pp. 1–46. János Bolyai Mathematical Society, Budapest, 1993.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning (ICML)*, 2023.

Kaushalya Madhawa, Katsuya Ishiguro, Kosuke Nakago, and Motoki Abe. GraphNVP: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Ayesh Madushanka, Renaldo T. Moura Jr., and Elfi Kraka. QM40, realistic quantum mechanical dataset for machine learning in molecular science. *Scientific Data*, 11:1376, 2024. doi: 10.1038/s41597-024-04206-y.

Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993. doi: 10.1109/78.258082.

Krzysztof Maziarz, Henry R. Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. In *International Conference on Learning Representations (ICLR)*, 2022.

Ali Moin, Andy Zhou, Abbas Rahimi, Abhishek Menon, Simone Benatti, George Alexandrov, Senam Tamakloe, Jonathan Ting, Natasha Yamamoto, Yasser Khan, Fred Burghardt, Luca Benini, Ana C. Arias, and Jan M. Rabaey. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. *Nature Electronics*, 4(1):54–63, 2021.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4602–4609, 2019.

Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. arXiv:1612.00005.

Igor Nunes, Mike Heddes, Tony Givargis, Alexandru Nicolau, and Alex Veidenbaum. Graphhd: Efficient graph classification using hyperdimensional computing. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1485–1490. IEEE, 2022. doi: 10.23919/DATE54114.2022.9774632.

Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995. doi: 10.1109/72.377968.

Prathyush Poduval, Haleh Alimohamadi, Ali Zakeri, Farhad Imani, M. Hassan Najafi, Tony Givargis, and Mohsen Imani. GrapHD: Graph-based hyperdimensional memorization for brain-like cognitive learning. *Frontiers in Neuroscience*, 16, 2022. doi: 10.3389/fnins.2022.757125.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*, 2025.

Abbas Rahimi, Simone Benatti, Pentti Kanerva, Luca Benini, and Jan M. Rabaey. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, 2016a.

Abbas Rahimi, Pentti Kanerva, and Jan M. Rabaey. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 64–69, 2016b.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. doi: 10.1038/sdata.2014.22.

Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences: Applying Deep Learning to Genomics, Microscopy, Drug Discovery, and More*. O'Reilly Media, 2019. ISBN 978-1492039839.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. arXiv:1906.00446.

Alpha Renner, Lazar Supic, Andreea Danielescu, Giacomo Indiveri, Bruno A. Olshausen, Yulia Sandamirskaya, Friedrich T. Sommer, and E. Paxon Frady. Neuromorphic visual scene understanding with resonator networks. *Nature Machine Intelligence*, 2024. arXiv:2208.12880.

Davide Rigoni, Nicolò Navarin, and Alessandro Sperduti. RGCVAE: Relational graph conditioned variational autoencoder for molecule design. *Machine Learning*, 114(2):47, 2025. doi: 10.1007/s10994-024-06638-4.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. doi: 10.1021/ci100050t.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. arXiv:2112.10752.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of GANs for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. arXiv:1907.10786.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: A flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations (ICLR)*, 2020.

Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Georgios Kaissis, Volker Tresp, and Bjoern Menze. Relationformer: A unified framework for image-to-graph generation. In *European Conference on Computer Vision (ECCV)*, pp. 422–439, 2022.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks (ICANN)*, volume 11139 of *LNCS*, pp. 412–422. Springer, 2018. doi: 10.1007/978-3-030-01418-6_41.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1908.01000.

Jonas Teufel, Luca Torresi, André Eberhard, and Pascal Friederich. Hyper-dimensional fingerprints as molecular representations, 2026. URL `https://arxiv.org/abs/2604.27810`.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research (TMLR)*, 2024. arXiv:2302.00482.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. arXiv:1711.00937.

Pere Vergés, Igor Nunes, Mike Heddes, Tony Givargis, and Alexandru Nicolau. Molecular classification using hyperdimensional graph classification. In *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024. doi: 10.1109/IJCNN60899.2024.10650746.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Scott A. Wildman and Gordon M. Crippen. Prediction of physicochemical parameters by atomic contributions. *Journal of Chemical Information and Computer Sciences*, 39(5):868–873, 1999. doi: 10.1021/ci990307l.

Robin Winter, Frank Noé, and Djork-Arné Clevert. Permutation-invariant variational autoencoder for graph-level representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018. doi: 10.1039/C7SC02664A.

Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. SimGRACE: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference*, 2022.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1810.00826.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018a.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *PMLR*, pp. 5708–5717, 2018b.

Chengxi Zang and Fei Wang. MoFlow: An invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 617–626. ACM, 2020. doi: 10.1145/3394486.3403104.

Xiaochen Zhang, Shuangxi Wang, Ying Fang, and Qiankun Zhang. MG-DIFF: A novel molecular graph diffusion model for molecular generation and optimization. *PLOS ONE*, 2025. doi: 10.1371/journal.pone.0331450.

Supplementary Material

CONTENTS

A Additional Background **18**
A.1 Random Walk Return Probabilities (RW) 18

B Extended Related Work **19**
B.1 Graph Autoencoders 19
B.2 Molecule-Specific Decoders 19
B.3 HDC for Graphs 20
B.4 Resonator Networks and HDC Decoding 21
B.5 Direct Graph Generation 21

C Molecular Generation Results **22**
C.1 Property-conditional generation 22
C.2 Distribution of Generated Molecules 23

D Experimental Details **25**
D.1 Datasets 25
D.2 Codec Hyperparameters 27
D.3 Codebook Construction 27
D.4 Iterative Edge Recovery 28
D.5 Greedy Beam Search Reconstruction 28
D.6 Bond-Order Post-Processing 29
D.7 Predictor Heads 29
D.8 Generation Pipeline 30
D.9 Compute 30
D.10 Software environment and code availability 30

E Ablation Studies **31**
E.1 Theoretical HRR capacity bound 31
E.2 Hypervector dimension D 31
E.3 Message-passing depth L 32
E.4 Beam width B_{beam} 32
E.5 RW bins B 32
E.6 RW walk-length set $\mathcal{K}$ 33
E.7 Vocabulary pressure as the unifying lever 33
E.8 Empirical decoding-time complexity 33

F Reconstruction Failure-Mode Breakdown **34**

G Additional Comparisons to SOTA Graph Autoencoders **35**

G.1 Compute Budget 36

G.2 Property Prediction 37

A Additional Background

This section collects textbook-style background material that is used in the main paper but not introduced there in detail. It is included for completeness and is not a contribution.

A.1 Random Walk Return Probabilities (RW)

The encoder of Sect. 4 of the main paper augments each node's feature tuple with a small vector of *random walk return probabilities* (RW) (Li et al., 2020; Dwivedi et al., 2022; Ma et al., 2023), a structural descriptor that captures local topology beyond what the node degree alone can express. This subsection defines the descriptor and traces its history in the recent graph-learning literature.

Definition. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ and diagonal degree matrix $\mathbf{D}$ with $D_{ii} = \sum_j A_{ij}$. The (uniform) random-walk transition matrix is $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$, so that M_{ij} is the probability that a walk currently at node i moves to node j in one step. Iterating, the entry $[\mathbf{M}^k]_{ij}$ gives the probability that a k-step random walk starting at i ends at j. The *k-step return probability* of node i is the diagonal entry

$$P_{\mathrm{rw}}^{(k)}(i) \;=\; [\mathbf{M}^k]_{ii}, \tag{A.1}$$

i.e. the probability that a k-step walk started at i has returned to i. For a fixed set of walk lengths $\mathcal{K} \subset \mathbb{N}$, the RW descriptor of node i is the vector $\big(P_{\mathrm{rw}}^{(k)}(i)\big)_{k\in\mathcal{K}} \in \mathbb{R}^{|\mathcal{K}|}$. Return probabilities are a classical object of study for random walks on graphs, going back at least to Lovász's survey of the area (Lovász, 1993).

What return probabilities encode. Return probabilities are sensitive to structural features that node degree alone cannot resolve: $P_{\mathrm{rw}}^{(2)}$ encodes the inverse-degree statistics of immediate neighbors. $P_{\mathrm{rw}}^{(k)}$ for larger even k responds to short cycles and fused-ring patterns through the closed walks they admit. Across $\mathcal{K}$ this gives a multi-scale fingerprint of local topology, which is known to lift the expressive power of message-passing GNNs and graph transformers above the ceiling of the 1-dimensional Weisfeiler–Leman test (1-WL) (Xu et al., 2019; Morris et al., 2019) on the substructures most relevant in molecular graphs (Li et al., 2020; Ma et al., 2023).

Prior work on random-walk-style descriptors. Li et al. (Li et al., 2020) introduce *Distance Encoding* (DE), in which the structural feature of a node u relative to a target set S is the collection of k-step landing probabilities from each $s \in S$ to u, with shortest-path distance and generalized PageRank admitted as special cases. Two architectures are proposed — DE as an extra node feature (DE-GNN) and DE as a controller of message aggregation (DEA-GNN) — and both are shown to be strictly more expressive than 1-WL on almost all r-regular graphs.

Dwivedi et al. (Dwivedi et al., 2022) introduce *Random Walk Positional Encoding* (RWPE) as the diagonal specialization of Li et al.'s pairwise construction: only the landing probability of node i to itself, $[\mathbf{M}^k]_{ii}$, is retained for each $k \in \{1,\dots,K\}$. The restriction reduces the per-graph storage cost from $\mathcal{O}(n^2K)$ to $\mathcal{O}(nK)$ and circumvents the sign-ambiguity of Laplacian eigenvector positional encodings, while preserving uniqueness of the node representation under mild conditions on the K-hop topology. RWPE coincides with the descriptor used by the encoder of Sect. 4 of the main paper.

Ma et al. (Ma et al., 2023) introduce the abbreviation *RRWP* (Relative Random Walk Probabilities) for the full pairwise tensor $\mathbf{P}_{i,j} = [\mathbf{I}, \mathbf{M}, \mathbf{M}^2, \dots, \mathbf{M}^{K-1}]_{i,j} \in \mathbb{R}^K$, used as a learned relative positional encoding in the GRIT graph transformer. They show that an MLP applied along the K-axis of RRWP can approximate, to arbitrary accuracy, K-truncated shortest-path distances and broad families of graph propagation matrices (sum and mean aggregation, K-truncated personalized PageRank, K-truncated heat kernels), thereby supplying a theoretical justification for the descriptor's expressive power. The diagonal slice $\mathbf{P}_{i,i}$ in their formulation coincides with RWPE.

The full pairwise tensor (DE, RRWP) and its diagonal restriction (RWPE) differ only in the off-diagonal entries. GRAFT uses the diagonal form for compactness, which we abbreviate as RW in the main paper.

B Extended Related Work

B.1 Graph Autoencoders

The reconstruction experiment of Sect. 5.1 of the main paper compares GRAFT to a small family of learned graph-level autoencoders that train an encoder–decoder pair end-to-end and evaluate exact reconstruction at the graph level.

GraphVAE (Simonovsky & Komodakis, 2018) is the first graph-level construction: a GCN encoder maps the input to a fixed-size latent and an MLP decoder produces a probabilistic adjacency tensor of fixed maximum size, supervised by an approximate graph-matching loss whose naïve cost is $\mathcal{O}(N^4)$ per training step. PIGVAE (Winter et al., 2021) replaces the matching with a jointly trained *permuter* that aligns input and decoded node orders, so reconstruction can be supervised in a fixed reference order; encoder and decoder are self-attention modules and the latent is a fixed-size order-invariant vector. GRALE (Krzakala et al., 2025) is the current state of the art among learned graph autoencoders and the primary reference point of the main paper. The architecture extends the Evoformer module of AlphaFold into a graph autoencoder by pairing it with a symmetric *Evoformer Decoder*, with both halves maintaining an explicit pair representation alongside the node representation that is coupled by row- and column-wise attention. The training objective combines an Any2Graph-style optimal-transport reconstruction loss with the learned-matching strategy of PIGVAE: a differentiable matching module is trained jointly with the encoder and decoder, predicting the assignment between input and reconstructed nodes and sidestepping the NP-complete inner optimization that the bare OT loss would entail. The pair representation makes the loss $\mathcal{O}(N^3)$ in compute and GPU memory, which caps the published GRALE experiments at $N = 32$ heavy atoms.

By contrast, GRAFT's encoder and decoder are constructed without any training. The matching problem that drives the cost of the learned baselines does not arise: the codebook lookup at decoding time pins each node back to its discrete feature tuple, the topology beam search recovers edges by their content rather than position, and there is no per-graph permutation to align.

B.2 Molecule-Specific Decoders

A second, larger family of molecular generative models sits adjacent to the graph-level autoencoders surveyed in Sect. B.1. These methods share the property that the decoder operates over a chemistry-aware vocabulary — ring-and-junction motifs, fragment libraries, or SMILES tokens — rather than over arbitrary attributed graphs.

JT-VAE (Jin et al., 2018) mines a vocabulary of ring and atom motifs from the training corpus and decodes a molecule in two stages, first the junction tree of motifs and then the atom-level assembly that resolves how each motif attaches at the cluster boundaries; validity is enforced architecturally by emitting only motifs from the mined vocabulary and only chemically legal attachments at assembly time. Its hierarchical successor HierVAE (Jin et al., 2020) extends this template to a multi-resolution motif hierarchy with attachment, motif, and atom layers, increasing the expressivity of the motif vocabulary at the cost of a more elaborate decoder. MoLeR (Maziarz et al., 2022) replaces the explicit junction tree with an autoregressive decoder that interleaves single-atom and motif extension steps conditioned on a partial graph and an optional scaffold, with the motif vocabulary learned from the training corpus and validity enforced by per-step valency masking; its construction targets scaffold-constrained generation rather than reconstruction and is evaluated on distribution metrics and scaffold completion rather than held-out graph isomorphism. ChemVAE (Gómez-Bombarelli et al., 2018) predates the graph-native decoders and operates entirely in SMILES space as a character-level encoder–decoder VAE with a property-prediction auxiliary head, with validity not enforced and the decoder routinely emitting ungrammatical strings; Grammar VAE (Kusner et al., 2017) addresses this by decoding a sequence of context-free-grammar production rules over the SMILES grammar, guaranteeing syntactic but not semantic validity.

All of these methods bake a molecular prior into the output space of the decoder: a junction-tree decoder operates over a mined motif vocabulary, a SMILES decoder over a fixed-length string grammar, and a fragment-extension decoder over an enumerated fragment library.

While GRAFT also uses prior chemistry domain information during its decoding process, it does not have to do so. For molecules in particular, the decoder is able to leverage the additionally available information about chemical valency rules, the number of implicitly attached hydrogens and the charge of an atom to further restrict the search space during decoding. However GRAFT's decoding algorithm in principle only requires the generically accessible information about the node

degree as a minimum requirement. We therefore mainly compare to the category of generic graph encoder–decoder systems introduced in the previous section — namely GraphVAE, PIGVAE and GRALE.

B.3 HDC FOR GRAPHS

Prior work on hyperdimensional computing for graphs splits along whether the encoder is paired with a decoder that recovers the input graph. The majority of the literature is encoder-only: an HDC mapping is constructed from binding and bundling primitives, and the resulting hypervector is fed to a similarity-based classifier or to a learned downstream head. Only one prior work pairs the encoder with a decoder.

Nunes et al.'s GraphHD (Nunes et al., 2022) establishes the encoder template that most subsequent work builds on to one extent or another. Each node is assigned a hypervector indexed by its PageRank-centrality rank, each edge (v_i, v_j) is encoded as the binding $\mathbf{x}_{v_i} \circledast \mathbf{x}_{v_j}$, and the resulting edge hypervectors are bundled into a single graph representation. GRAFT's round-1 readout (Eq. 3 of the main paper) reuses this bind-and-bundle primitive, but indexes nodes by their feature tuple through a shared codebook so that decoded endpoints carry the same chemical identity across graphs rather than a per-graph centrality position.

Subsequent encoder-only methods retain the same primitive and vary the encoding scheme. MolHD (Vergés et al., 2024) replaces centrality-rank indexing with a star-subgraph readout that binds each node with all of its neighbors before bundling, paired with the RefineHD adaptive-classification head and evaluated on PubChem anticancer screen classification. HDBind (Jones et al., 2024) works upstream of the graph entirely, converting pre-extracted ECFP and MoLFormer fingerprints into hypervectors via random projection for similarity-based virtual screening; the molecular graph itself is not encoded. The hyperdimensional fingerprints (HDF) of Teufel et al. (Teufel et al., 2026) are the most direct precedent for GRAFT's encoder: atomic features are mapped to hypervectors via predefined dictionaries, structural information is propagated through L rounds of bind-and-bundle message passing, and the per-iteration node hypervectors are aggregated into a single fingerprint, evaluated on molecular property prediction and similarity search. The remaining encoder-only HDC graph methods address node-level rather than graph-level tasks. RelHD (Kang et al., 2022) extends Nunes et al.'s GraphHD to multi-hop relational node classification by maintaining per-hop neighbor bundles $\mathbf{H}_i^{(k)}$ and combining them with a node-feature term via role-marker bindings. HDGL (Dalvi & Honavar, 2025) replaces RelHD's binary indicator features with a random-hyperplane LSH map of continuous node features and uses permutation-protected binding to recover WL-style injectivity in the aggregation, evaluated on transductive node classification and link prediction. HyperGraphX (Cong et al., 2025) fuses HDC primitives with GCN-style message passing through a weightless degree-normalized aggregation and an input/aggregated residual mix, again evaluated on transductive node classification. Ultimately, none of these methods aims to recover the original graph structure from its encoded form.

The only prior method that pairs the HDC encoder with a decoder is Poduval et al.'s GrapHD[1] (Poduval et al., 2022), whose encoder bundles pairwise neighbor bindings in a manner algebraically close to GRAFT's edge term. Its decoder, however, does not transfer to the attributed-graph autoencoder setting addressed by GRAFT: each graph is encoded with a fresh random draw of per-node hypervectors and recovered by iterative noise cancellation against that draw, so the same chemical identity is assigned a different hypervector in every graph — whereas a corpus-level autoencoder requires that a given atom type decode to the same species across all instances. The construction additionally operates on graph topology alone, with no facility for the discrete node or edge attributes that an attributed-graph decoder must recover exactly, and is evaluated only on small synthetic random graphs.

GRAFT differs from this body of work in two ways. The encoder iterates the bind-and-bundle primitive for L rounds of message passing, augmenting the per-round bound-pair bundle with a separate node-population term that retains node-level information needed at decoding time, and lifts expressiveness toward the L-iteration 1-WL ceiling. The decoder of Sect. 4 of the main paper recovers attributed graphs from a single hypervector through a shared feature-tuple codebook and a topology beam search, assigns a consistent identity to each atom type across the entire dataset, and handles discrete node and edge labels exactly.

[1] Distinct from Nunes et al.'s GraphHD discussed above; the two works share a near-identical name but no overlap in authorship, encoder, or evaluation.

B.4 Resonator Networks and HDC Decoding

The edge-recovery stage of GRAFT factors a superposition $\sum_k \mathbf{x}_{a_k} \circledast \mathbf{x}_{b_k}$ into its constituent pairs given a known codebook $\{\mathbf{x}_v\}$, an extension of the canonical *resonator-network* setting — factoring a single bound product $\mathbf{x}_a \circledast \mathbf{x}_b$ into its factors (Frady et al., 2020; Kent et al., 2020; Renner et al., 2024) — to a sum of such products, as studied for VSAs by Hersche et al. (2023a). A resonator network maintains a separate estimate per factor slot and updates each slot by reciprocal binding against the composite vector, projecting back onto the corresponding codebook at every step; the dynamics interleave VSA operations with pattern completion and converge to a factor decomposition by searching in superposition over all candidate combinations (Frady et al., 2020; Kent et al., 2020). Recent work has also demonstrated resonator factorization on neuromorphic hardware (Renner et al., 2024). GRAFT instead extracts edge atoms one at a time by greedy correlation against the precomputed pairwise codebook and subtracts each recovered atom from a running residual, an instance of the matching-pursuit algorithm of Mallat and Zhang (Mallat & Zhang, 1993). The choice of matching pursuit over resonator iteration is pragmatic rather than principled: in our regime edge recovery is empirically near-saturating and runs in negligible time relative to the topology beam search of the main paper's decoder, so both the dominant compute cost and the dominant residual error of GRAFT sit downstream of edge recovery, in the beam-search stage that pairs endpoint identities with edge atoms (Sect. 4 of the main paper). Therefore, we did not run an empirical comparison against a resonator-network edge decoder: any change at this stage would alter neither the dominant cost nor the dominant residual error of GRAFT's decoder, and so could not move the bottleneck identified above. Resonator networks would, however, be a natural consideration should edge decoding itself become a limiting factor for larger graphs in the future.

B.5 Direct Graph Generation

Most prior approaches to graph generation, particularly for molecular graphs, operate directly on the graph rather than through a fixed graph-level representation as GRAFT does. We briefly survey this landscape here to situate the unconditional ZINC250K experiment of the main paper.

Older approaches fall into a handful of families: variational autoencoders learn an explicit latent over molecular graphs (GraphVAE (Simonovsky & Komodakis, 2018), JT-VAE (Jin et al., 2018)); autoregressive models emit one node or edge at a time (GraphRNN (You et al., 2018b), MolGPT (Bagal et al., 2022)); normalizing-flow models invert a learned bijection over node and adjacency tensors (GraphNVP (Madhawa et al., 2019), MoFlow (Zang & Wang, 2020), GraphAF (Shi et al., 2020)); reinforcement-learning and gradient-free search methods treat generation as reward-driven optimization (GCPN (You et al., 2018a), GraphGA (Jensen, 2019)).

The recently dominant paradigm of diffusion-style and flow-matching generative models on the discrete graph sets the present state of the art on molecular benchmarks. GDSS (Jo et al., 2022) introduces continuous-time score-based diffusion for graphs through a joint node-and-adjacency SDE with permutation-equivariant partial scores. DiGress (Vignac et al., 2023) reformulates the same problem as discrete denoising diffusion using marginal-preserving Markov transitions over categorical node and edge types, paired with a graph-transformer denoiser, and reports strong results on QM9, MOSES, and GuacaMol. DisCo (Xu et al., 2024) and MG-Diff (Zhang et al., 2025) extend the discrete-diffusion setup with alternative corruption schedules — continuous-time absorbing-state diffusion and a mask-and-replace process with graph padding, respectively. DeFoG (Qin et al., 2025) recasts the setting as discrete flow matching with a continuous-time Markov-chain denoiser and a linear-interpolation noising process, reaching state-of-the-art molecular generation with 5–10% of the sampling steps used by diffusion baselines.

Most of these methods share a common limitation around graph size: the number of nodes N is not part of the generative model itself, but is sampled separately from a size prior — typically just the empirical size distribution of the training set — and then held fixed for the rest of the generation. The codec-latent flow matching of GRAFT avoids this step entirely because the codec hypervector already encodes N implicitly: the bound-pair bundle accumulates one term per edge and the node-population term one term per node, so the graph size is simply read off the sampled hypervector at decoding time.

C Molecular Generation Results

In addition to graph reconstruction and property prediction, we also conduct initial experiments on graph generation by using the HDC space, or a compressed latent thereof, as the target for a simple generative model. We use a continuous-time flow matching model (Lipman et al., 2023) to transport samples of a standard isotropic Gaussian source distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to the target distribution.

C.1 Property-conditional generation

Setup Aside from the unconditional generation we also conduct an initial experiment on property-conditional generation for the GRAFT+AE scenario — reusing the pre-trained autoencoder from the unconditional experiments. We use the principle of classifier-free guidance (Ho & Salimans, 2021) to train a flow model conditioned on the Crippen LogP value (Wildman & Crippen, 1999), calculated for all ZINC samples by RDKit. This is done by concatenating an embedding of each sample's ClogP target value to the time embedding entering the AdaZero blocks as a condition. Subsequently during training, on 90% of the samples the flow is trained with the additional conditional information of the target property. The remaining 10% of the samples are trained with a special fixed unconditional token $\varnothing$, which replaces the property embedding and signals the absence of a conditioning value. At inference time, classifier-free guidance combines the conditional and unconditional velocity predictions of the same network into a single guided velocity that is then integrated by the ODE solver. Concretely, at each integration step we evaluate the velocity network twice — once with the target condition c and once with the unconditional token $\varnothing$ — and form

$$\tilde{\mathbf{v}}_\theta(\mathbf{x}_t, t, c) \;=\; (1 + w)\, \mathbf{v}_\theta(\mathbf{x}_t, t, c) \;-\; w\, \mathbf{v}_\theta(\mathbf{x}_t, t, \varnothing), \tag{C.2}$$

where $w \geq 0$ is the guidance weight. Setting $w = 0$ recovers the purely unconditional flow, whereas larger w extrapolates the velocity along the conditional–unconditional difference and increases the strength with which the target property is enforced, at the cost of reduced sample diversity.

Discussion We intentionally choose the ClogP value as the target property because it is both (i) easily and deterministically computable and (ii) known to strongly correlate with the graph size, where higher ClogP values can usually be found in larger molecules and vice versa.
In this experiment, GRAFT is able to accurately match the mean of both target ClogP values at the very edges of the dataset distribution (-1, +5), albeit with a relatively large spread (see Fig. C.1). More importantly, both experiments showcase a substantial deviation in the distribution of the graph size, showing that the flow model was able to successfully exploit this known correlation.

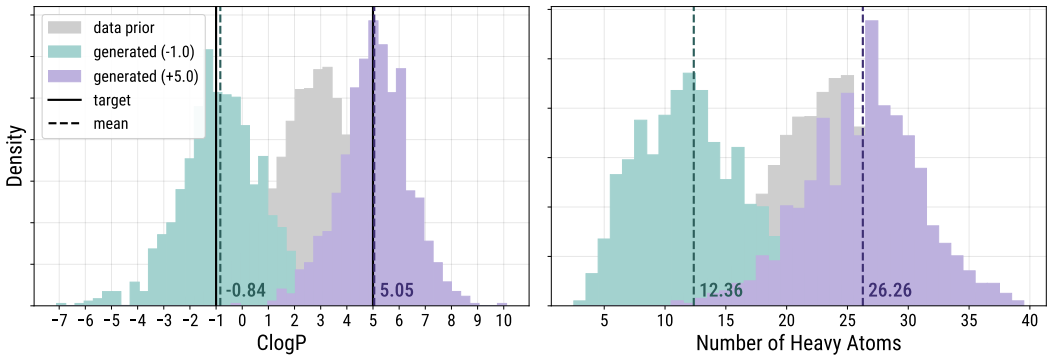


Figure C.1: Property-conditional generation with classifier-free guidance for two ClogP targets at the edges of the ZINC250K distribution. Both target conditions are overlaid in each panel (teal: target -1; purple: target $+5$) on top of the ZINC250K reference (gray). Left: RDKit-computed ClogP values across $n = 1000$ generated molecules per target; solid vertical lines mark the conditioning targets, dashed lines mark the empirical means of the generated samples. Right: heavy-atom count of the generated molecules.

While the overall quality of GRAFT-based molecular generation does not yet match that of SOTA discrete diffusion and flow matching models (Vignac et al., 2023; Qin et al., 2025), we see concrete advantage in this property of capturing graph size and topology at the same time in a single unified representation. Modern graph generation approaches have to split this into a two-step procedure: First, the graph size has to be sampled from a reference distribution (most often the dataset distribution) such that the subsequent denoising process can operate on this pre-determined size. This approach further complicates property-conditional generation where the property correlates with the graph size. We believe that further improvements of the GRAFT methodology and GRAFT-based graph generation could provide a competitive and principled alternative to this two-step paradigm.

C.2 Distribution of Generated Molecules

Distributional Misalignment. While the graph generation in Sect. 5.4 of the main paper shows good results in terms of validity, uniqueness and novelty, there is still a noticeable gap in the distributional alignment with respect to the dataset statistics. In the main text this is shown primarily for the quantitative estimate of drug-likeness (QED) and the synthetic accessibility score (SAS) as they are calculated by RDKit. However, the same can be observed for other simple metrics (see Fig. C.2). While the sampled molecules show good overlap with the training distribution on purely statistical metrics such as the atom and edge type distributions, they deviate more strongly on topological properties, such as the max ring size and ring count distributions. In terms of the computed properties, the largest deviation can be observed for the SAS — which aligns with the previous observation as the score heavily penalizes unusual ring structures.

Generation Failures. When looking at individual generations (see Fig. C.3) we can observe this pattern more concretely. A part of the generated molecules look visually realistic, in that they contain common organic fragments that can be found in the training dataset as well, and are composed of common 5- and 6-membered aromatic ring structures.
Other samples on the other hand exhibit two dominant failure modes specifically related to ring structures. Some molecules consist of long carbon chains which never properly close into a ring at all, while others form unrealistically large ring structures of 20+ atoms instead separating these into more likely structures of multiple individual 5- or 6-membered rings. Investigating these cases, we identify multiple possible reasons for these failures:

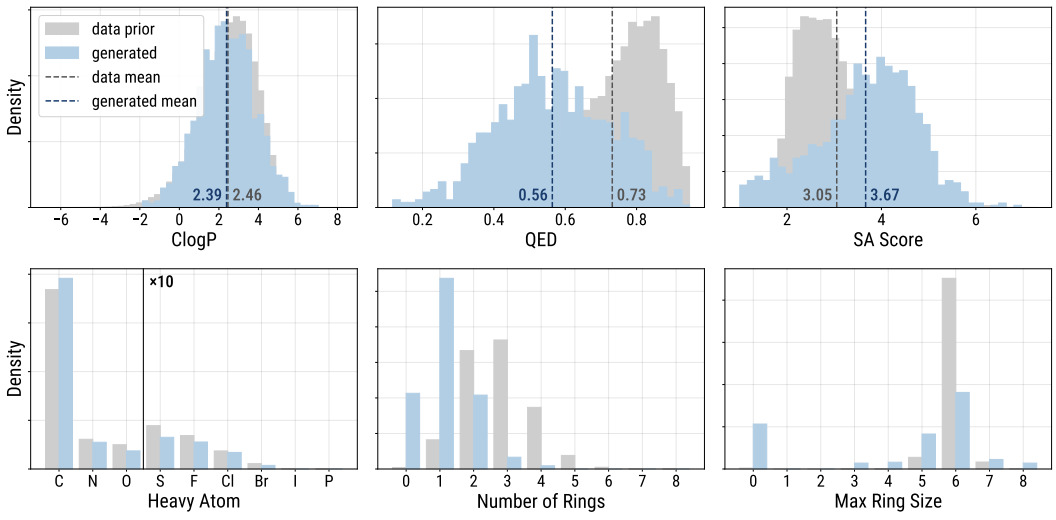


Figure C.2: Distributional comparison between unconditional GRAFT-generated molecules and the ZINC250K reference distribution across simple structural and computed-property metrics: atom-type frequencies, bond-type frequencies, heavy-atom count, ring count, maximum ring size, QED, and SAS.

- The beam search underlying the molecule decoding is based on the information about the degree, number of hydrogens and the charge contained within the feature tuple of an atom to identify those nodes inside the partially constructed graphs which can still accept more edges. When generating samples, the resulting edge set that is retrieved is not entirely self-consistent, meaning that even if all of the used edges are used up, there are still remaining residuals — indicating that the total sum of existing edges and edges expected according to the node degree information don't fully match.
- Another problem that can arise from the inconsistent edge sets is that some reconstructed graphs are not able to use the edges that were originally retrieved from the edge embedding. Rather than leaving open residuals, it can also happen that available residuals are used up before all of the edges have been inserted into the partial graph. Since the decoding algorithm respects chemical valency rules, the left over edges are discarded rather than forcefully inserted into the graph. This in turn can lead to samples with unclosed ring structures.
- Finally, the imperfect generation also causes a misalignment of the edge and graph terms in the dual stack representation. While perfect reconstructions exit the decoding with a cosine similarity of 1.0 between the given HDC vector and the HDC vector computed from the best search candidate, this final similarity lies between 0.6 to 0.9 for generated samples — indicating that no perfect match could be found. This additional source of noise further contributes to the distortion of samples.

Discussion. To further improve the graph generation results in the future, the sources of this distributional misalignment and failure modes will have to be identified and addressed.
We believe one primary reason for the difficulty with the generation of the molecules is the nature of the HDC-based embedding space in general. Since repeated binding operations produce outcomes which are again quasi-orthogonal to the inputs, the resulting space resembles a widely distributed collection of isolated quasi-discrete valid points, separated by large stretches of mostly invalid space. Such a distribution should present a target that is particularly hard to learn for a flow model using an isotropic Gaussian distribution as a source.
This is further supported by the improvements gained from applying an autoencoder compression to this HDC space prior to the sampling process. For the ZINC dataset, a standard autoencoder architecture proved to still have a lot of potential for improvement — saturating at approximately 60% GI accuracy for encoder-to-decoder graph reconstruction. We expect the hypervector autoencoder to benefit from the same recipe that has driven the success of latent generative models in the image domain (Esser et al., 2021; Rombach et al., 2022): decoupling perceptual compression from generative modeling by training a single high-capacity autoencoder once on a much larger corpus

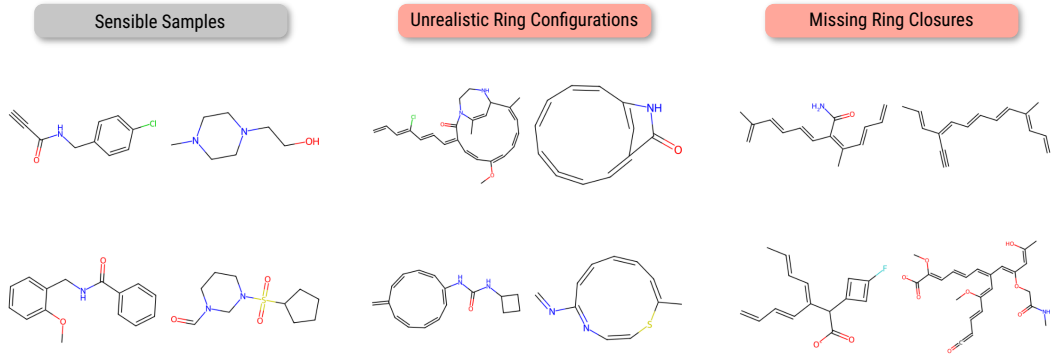


Figure C.3: Sample plate of 2D structures drawn at random from the unconditional GRAFT generations on ZINC250K. Left: visually realistic samples composed of common organic fragments and standard 5-/6-membered aromatic ring systems. Middle: representative samples illustrating uncommon ring configurations. Right: representative samples illustrating samples missing ring closing edges.

and reusing its frozen latent space across downstream generation tasks, rather than retraining a small AE end-to-end with each generative model.
Another challenge for the quality of the generation results arises from the limited self-consistency of the generated samples — both within the retrieved edge set and the interplay of edge and graph term in the dual stack representation. We believe this problem gets amplified by the size, and therefore edge, vocabulary as it becomes harder to generalize rules of connectivity across many more seemingly distinct node types. In the current GRAFT approach, the random walk features are the main drivers of vocabulary size — intentionally included to substantially improve reconstruction performance. For the task of graph generation these random walk features present a particular tension: Including them substantially increases the complexity of the target distribution but omitting them would decrease the basic reconstruction accuracy itself, leading to sub-optimal generation outcomes either way. Fully addressing this issue will most likely require an improvement of the decoding algorithm such that fewer or no random walk features are required to achieve similar reconstruction accuracy.

D Experimental Details

This appendix collects the per-experiment training and evaluation details referenced but not fully specified in the main paper, starting with the datasets and hyperparameters, followed by additional algorithm descriptions and details on the learned components.

D.1 Datasets

GRAFT is evaluated on five molecular-graph datasets across the four experiments of Sect. 5 of the main paper. Tables D.1 and D.2 collect the size and split information. Subsequent paragraphs document preprocessing, atom-type vocabulary, and per-experiment usage.

PubChem-16 / 32 / 64. The three PubChem reconstruction benchmarks are assembled directly from the PubChem Compound database (Kim et al., 2016) by downloading the raw SMILES tables and applying the heavy-atom cap and filtering rules described below. Molecules with disconnected components, with parser failures, or containing atoms outside the vocabulary $\{$B, Br, C, Cl, F, I, N, O, P, S, Si, Sn$\}$ are dropped at load time. An 80/10/10 train/valid/test split is assigned deterministically by hashing each PubChem compound identifier (CID), so molecules that pass the cap on all three variants fall in the same split across variants. For each variant, a fixed pool of $250\,000$ molecules is sampled once and reused across runs as the codebook-construction pool from which the codec reads its set of observed feature tuples.

QM9. The QM9 dataset used for Fingerprint2Graph task was introduced by Ramakrishnan et al. (Ramakrishnan et al., 2014). The atom-type vocabulary is $\{$C, N, O, F$\}$. Molecules with disconnected fragments and molecules whose SMILES parser call fails are dropped at load time. After filtering, 99.8% of the original SMILES remain.
For Fingerprint2Graph, each molecule is represented by a Morgan circular fingerprint of radius 2 (ECFP4) hashed to a $2\,048$-bit vector via the standard RDKit fingerprint generator, with chirality and pharmacophore-feature flags both disabled (the RDKit defaults).

GDB13 ABCDEFGH subset. The underlying GDB-13 enumeration of Blum and Reymond (Blum & Reymond, 2009) contains all $\sim$977 million organic molecules of up to 13 heavy atoms drawn from $\{$C, N, O, S, Cl$\}$ that pass simple chemical-stability and synthetic-feasibility rules. Blum, van Deursen, and Reymond (Blum et al., 2011) subsequently defined eight cumulative drug-likeness and scaffold-likeness filters labeled A through H (Table 2 of that paper, e.g. the "rule of three" fragment-likeness filter and a "no acyclic carbon atoms" scaffold-likeness filter). The ABCDEFGH variant is the most restrictive of these subsets, retaining $1\,470\,284$ molecules after all eight filters have been applied, and is the variant we use. We download it from the authors' Zenodo release at `https://zenodo.org/record/5172018/files/GDB13_Subset-ABCDEFGH.smi.gz`. Fingerprint construction matches the QM9 setup (Morgan circular fingerprints of radius 2 hashed to $2\,048$ bits, with chirality and pharmacophore-feature flags disabled).

MoleculeNet. The seven property-prediction tasks (QM9, QM40, ESOL, LIPO, FreeSolv, BBBP, BACE) are taken from the MoleculeNet benchmark (Wu et al., 2018). The six standard tasks come from the DeepChem library (Ramsundar et al., 2019) and QM40 is downloaded from its Figshare release at `https://doi.org/10.6084/m9.figshare.25993060`. Each task is split into

Table D.1: Reconstruction datasets (§5.1). Heavy-atom statistics are computed over the 250 000-molecule codebook-construction pool of each variant. The same hash partition (App. D.3) is applied globally across all three variants, so the three pools are nested by atom-count cap and not independent draws.

Dataset	Total	Atoms / molecule (mean ± std, max)	Train pool	Eval pool/seed
PubChem-16	∼14M	$13.248 \pm 2.570,\ 16$	250 000	10 000
PubChem-32	∼84M	$18.944 \pm 6.155,\ 32$	250 000	10 000
PubChem-64	∼118M	$20.806 \pm 8.955,\ 64$	250 000	10 000

Table D.2: Downstream-task datasets. Heavy-atom statistics are computed over the loader-filtered training partition. "Num. Elements" is the total dataset size (train + valid + test). "Split protocol" specifies the train/valid/test partitioning used for that dataset. "Vocab. Size" is the cardinality of the hypernet's atom-type vocabulary, marked † for the ZINC encoder vocabulary {Br, C, Cl, F, I, N, O, P, S} and ‡ for the PubChem encoder vocabulary {B, Br, C, Cl, F, I, N, O, P, S, Si, Sn}.

Dataset	Num. Elements	Split protocol	Atoms / molecule		Vocab. Size
			mean ± std	max	
QM9	133 376	random 80/10/10	8.8 ± 0.5	9	$9^{\dagger}$
GDB13 (ABCDEFGH)	1 470 284	random 80/10/10	12.8 ± 0.6	13	$9^{\dagger}$
ZINC250K	249 456	pre-shipped 88/10/2	23.2 ± 4.5	38	$9^{\dagger}$
QM9 (12 targets)	133 885	random 80/10/10	8.8 ± 0.5	9	$12^{\ddagger}$
QM40	162 954	random 80/10/10	23.3 ± 7.7	37	$12^{\ddagger}$
ESOL	1 128	random 80/10/10	13.3 ± 6.9	55	$12^{\ddagger}$
LIPO	4 199	random 80/10/10	27.0 ± 7.4	115	$12^{\ddagger}$
FreeSolv	642	random 80/10/10	8.7 ± 4.2	24	$12^{\ddagger}$
BBBP	1 934	random 80/10/10	23.8 ± 10.4	132	$12^{\ddagger}$
BACE	1 513	random 80/10/10	34.1 ± 8.5	97	$12^{\ddagger}$

80/10/10 train/valid/test partitions at random, independently across the three reported seeds.
We adopt two dataset-level choices from the GRALE evaluation protocol (Krzakala et al., 2025) that shape the slice each task is run on. First, following (Krzakala et al., 2025, App. A.1), molecules with more than $N = 32$ heavy atoms are dropped before training and evaluation, so the comparison in Table 3 of the main paper runs on the resulting truncated release of each task. Second, for QM9 and QM40 — both of which ship with multiple regression targets — we follow (Krzakala et al., 2025, footnote 4) and train and evaluate on the internal-energy target only.

ZINC250K. We use the standard ZINC250K drug-like subset of the ZINC database (Irwin et al., 2012), with an 80/10/10 train/valid/test split. The atom-type vocabulary is {Br, C, Cl, F, I, N, O, P, S}. Disconnected and unparseable SMILES are dropped at load time. The QED and synthetic-accessibility scores reported in Table 4 of the main paper are computed via RDKit's standard QED implementation and the synthetic-accessibility-score routine of the RDKit Contrib package, respectively. The reference statistics $\mu_{\text{QED}} = 0.73$ and $\mu_{\text{SAS}} = 3.05$ used in the ΔQED and ΔSAS columns are computed on the full ZINC250K release.

General preprocessing. All molecular loaders share the following conventions, which apply to every row of Tables D.1 and D.2.

(i) *RDKit version.* We use RDKit Version 2025.9.5 for all our preprocessing and property computations.

(ii) *Sanitization.* SMILES strings are parsed with RDKit under its default sanitization rules. For chemistry-level evaluation (App. D.6), the graph-to-molecule reconstruction additionally enforces explicit sanitization and a kekulé bond assignment.

(iii) *Hydrogens.* Hydrogens are kept implicit throughout: the loaders neither add nor remove hydrogen atoms, and each atom's total hydrogen count is captured as a node feature.

Table D.3: Headline configuration of GRAFT for the PubChem reconstruction runs of Sect. 5.1 of the main paper. The same configuration is used across PubChem-16, PubChem-32, and PubChem-64.

Setting	Value
Encoder	
Hypervector dimension D	1024
Message-passing depth L	5
RW walk-length set $\mathcal{K}$	$\{4, 8, 12, 18\}$
RW bins per walk length B	8
VSA model	HRR (circular convolution)
Per-round normalization	per-node ℓ_2 at each round $l \geq 1$
Decoder	
Beam width B_{beam}	32
Initial prune threshold T_0	32
Sustained prune threshold T_1	32
Numerics	
Floating-point precision	FP64 (double)
Evaluation protocol	
Codebook draw seed	42
Eval-subsample seeds	$\{42, 43, 44\}$
Eval subsample size	10 000 molecules / seed
Stratification	heavy-atom-count quartile
Train pool size	250 000 molecules / variant

(iv) *Stereochemistry.* Per-atom chirality and per-bond stereo information are not retained as graph features, so E/Z and R/S information does not influence the encoded graph. The canonical SMILES string stored alongside the graph (used only for chemistry-level evaluation) preserves stereo markers.

(v) *Disconnected components.* Molecules whose SMILES indicates more than one disconnected component are dropped at load time across all datasets.

D.2 Codec Hyperparameters

The reconstruction numbers in the main paper's Sect. 5.1 reconstruction table all use a single *headline configuration* of GRAFT across PubChem-16, PubChem-32, and PubChem-64; Table D.3 collects the relevant hyperparameters fixed for these runs.

D.3 Codebook Construction

Node codebook. In the GRAFT encoding scheme, each node v_i is associated with a discrete feature tuple $\mathcal{X}_i = (x_1^i, x_2^i, \ldots, x_n^i)$. We call the mapping $\mathbf{C}_{\text{node}}$ of these tuples to randomly drawn hypervectors $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \frac{1}{D}\mathbf{I}_D)$ the *node codebook*. Since every unique combination of discrete properties is assigned its own vector representation, the size of this codebook can become very large as the number of individual discrete properties and their respective number of possible states increases. Particularly when using the additional random walk features, the size of the codebook can become intractable. However, the size of the codebook can be decreased significantly as not every feature combination can or will appear in practical settings. We therefore define the *realized vocabulary* $\mathcal{X}^\star$ as the subset of the full Cartesian product of feature components consisting only of the observed feature combinations, and restrict the codebook to a table $\mathbf{C}_{\text{node}} \in \mathbb{R}^{|\mathcal{X}^\star| \times D}$ over its rows.
A general version of such a realized vocabulary can be obtained by applying the fundamental rules of chemistry: a carbon atom for instance has only 4 valence electrons and therefore cannot have a node degree higher than 4 at neutral charge. Similar rules can be defined for all possible combinations of atom type, node degree, charge and number of implicit hydrogens — thereby substantially decreasing the overall codebook's size. In our experiments we further reduce the codebook size by keeping in memory only the feature combinations actually observed in the respective datasets.

Edge codebook. The matching-pursuit edge decoder (App. D.4) repeatedly ranks pairwise codebook bindings against a residual, so we precompute these bindings once as the *edge codebook* $\mathbf{C}_{\text{edge}} =$

Algorithm 1 Matching-pursuit edge recovery.

Require: Edge term $\mathbf{h}_{\text{edge}}$, edge codebook $\{\mathbf{c}_p = \mathbf{x}_a \circledast \mathbf{x}_b\}_{p\in\mathcal{P}}$ over ordered observed-edge type pairs $p = (a, b) \in \mathcal{E}^\star$, threshold ε, per-dataset cap $E_{\max}$

Ensure: Decoded edge-type multiset $\hat{\mathcal{E}}$

$\mathbf{r} \leftarrow \mathbf{h}_{\text{edge}}$; $\hat{\mathcal{E}} \leftarrow [\,]$; $r_{\text{prev}} \leftarrow \|\mathbf{r}\|_2$
while $|\hat{\mathcal{E}}| \le E_{\max}$ **do** ▷ Halt 1: edge cap reached
  $p^\star = (a^\star, b^\star) \leftarrow \arg\max_{p\in\mathcal{P}} \cos(\mathbf{r},\ \mathbf{c}_p)$
  $\mathbf{r} \leftarrow \mathbf{r} - (\mathbf{x}_{a^\star} \circledast \mathbf{x}_{b^\star} + \mathbf{x}_{b^\star} \circledast \mathbf{x}_{a^\star})$
  if $\|\mathbf{r}\|_2 \le \varepsilon$ **then break** ▷ Halt 2: residual exhausted
  end if
  if $\|\mathbf{r}\|_2 > r_{\text{prev}}$ **then break** ▷ Halt 3: residual stopped decreasing
  end if
  $r_{\text{prev}} \leftarrow \|\mathbf{r}\|_2$
  $\hat{\mathcal{E}} \leftarrow \hat{\mathcal{E}} \cup \{(a^\star, b^\star),\ (b^\star, a^\star)\}$
end while
return $\hat{\mathcal{E}}$

$\{\mathbf{x}_a \circledast \mathbf{x}_b : (a, b) \in \mathcal{E}^\star\}$, where $\mathcal{E}^\star \subseteq \mathcal{X}^\star \times \mathcal{X}^\star$ is the set of ordered feature-tuple pairs observed as edges in the codebook-construction pool. Because binding is non-commutative, the encoding of an undirected edge in Eq. 3 of the main paper picks up both ordered bindings — one from each endpoint's neighbor bundle — so each edge enters $\mathbf{h}_{\text{edge}}$ as the symmetric sum $\mathbf{x}_a \circledast \mathbf{x}_b + \mathbf{x}_b \circledast \mathbf{x}_a$. The decoder accordingly subtracts both orientations from the residual per accepted edge, and records both (a, b) and (b, a) in its output multiset (App. D.4).

D.4 Iterative Edge Recovery

Algorithm 1 specifies the matching-pursuit edge-recovery loop. Each accepted edge subtracts both binding orientations $\mathbf{x}_a \circledast \mathbf{x}_b$ and $\mathbf{x}_b \circledast \mathbf{x}_a$ from the residual, matching the symmetric sum that the encoder bundles for that edge (App. D.3). Specifically, the edge term $\mathbf{h}_{\text{edge}}$ is left un-normalized in Eq. 3, since rescaling the input by $1/\|\mathbf{h}_{\text{edge}}\|_2$ would add noise to the edge recovery process. The per-round normalization is therefore only applied to the readout of the graph term $\mathbf{h}_{\text{graph}}$ to prevent the layerwise increase of the vector norm caused by the repeated binding.

The loop halts the first time any of three conditions is met:

1. The decoded multiset reaches a upper bound $E_{\max}$.
2. The residual ℓ_2 norm drops below a fixed threshold ε, set to one percent of the smallest per-edge step size over the codebook.
3. The residual norm fails to decrease relative to the previous iteration, indicating that the next subtraction would amplify rather than reduce noise.

This threshold is calibrated against the fixed-magnitude residual chunk that an accepted edge removes; the encoder accordingly leaves $\mathbf{h}_{\text{edge}}$ un-normalized in Eq. 3 of the main paper, since rescaling the input by $1/\|\mathbf{h}_{\text{edge}}\|_2$ would invalidate the calibration. The per-round normalization in $\mathbf{h}_{\text{graph}}$ (Eq. 4 of the main paper) is a separate operation on the graph-term readout and does not interact with the edge recovery.

D.5 Greedy Beam Search Reconstruction

The dual-stack readout $\mathbf{h}$ assigns each term one decoding role: $\mathbf{h}_{\text{edge}}$ supplies the edge inventory, recovered by the matching pursuit (App. D.4), and $\mathbf{h}_{\text{graph}}$ guides the beam-search reconstruction of the graph topology.

The matching pursuit returns an edge-type multiset $\hat{\mathcal{E}} \subseteq \mathcal{X}^\star \times \mathcal{X}^\star$ that lists the types of all recovered edges but not how they connect: two edges of types (a, b) and (a, c) could attach to a shared a-typed node, or to two separate a-typed nodes, and both assemblies are equally consistent with $\hat{\mathcal{E}}$. The second decoding stage enumerates the candidate graphs compatible with $\hat{\mathcal{E}}$ and ranks them against

$\mathbf{h}_{\text{graph}}$.
We perform this enumeration by a greedy beam search that grows partial graphs one *node* at a time from single-edge seeds, scoring each candidate G' as $s(G') = \cos(\mathbf{h}_{\text{graph}}, \mathbf{h}_{G'})$, where $\mathbf{h}_{G'}$ is the graph term obtained by re-encoding G'. Importantly, when re-encoding a partial graph G' we dont re-derive the node features like the degree, number of hydrogens or the random walk features from the partial state but rather use the known "ground truth" feature tuples that have already been recovered from the set of edge terms. This ensures that the encoding of the partial graph is consistent with the given full-graph encoding.
Each iteration extends every current partial graph G' by adding one new node, attaching it via at least one edge to a chosen *anchor* (a placed node that can still accept more edges), and optionally also via further edges to other already-placed nodes — these additional attachments are what let the search assemble rings. We seed the search with all single-edge graphs $(a, b) \in \hat{\mathcal{E}}$ and extend each seed until it reaches the target node count. At every step the anchor is chosen as the node with the lowest *residual degree* — the number of edges it can still accept, computed as the target bond count implied by its feature tuple (atom type, formal charge, and number of hydrogens) minus the edges already placed at it in G'. Picking the lowest-residual anchor keeps the per-step branching factor bounded without ruling out any reachable graph. Duplicate partial graphs arising from different extension orderings or from the two orientations of the same seed edge are collapsed by a graph-isomorphism hash.

D.6 Bond-Order Post-Processing

The decoder returns a graph $\hat{G}$ with node attributes and unweighted edges. Molecular evaluation requires real bond orders, which we assign in a small post-processing pass that balances each atom's valence against its decoded environment.

For each atom i, we compute a target valence V_i from the atom type, formal charge q_i, and hydrogen count h_i that the codec already recovered; for multi-valent elements (e.g. S, N, halogens) the target is the smallest standard valence consistent with the atom's heavy-atom degree, refined against the valence demand of its one-hop neighbors. All bonds start at order 1 and are greedily upgraded whenever both endpoints still carry an unmet valence demand, up to a cap of 3, with a short follow-up pass that recovers the aromatic-ring orderings the greedy upgrade misses. RDKit then maps the integer bond orders to the corresponding bond types, sanitizes the molecule, and emits the final SMILES string.

D.7 Predictor Heads

MoleculeNet property prediction. For each MoleculeNet task we attach a SwiGLU residual MLP head on top of the frozen $2D$-dimensional dual-stack hypervector $\mathbf{h} = [\mathbf{h}_{\text{edge}} \,\|\, \mathbf{h}_{\text{graph}}]$. All hypervectors are precomputed once with the headline codec configuration and cached, so only the head trains. The head consists of a stack of projection blocks, each being a linear projection at the configured width followed by a small number of SwiGLU residual blocks at the same width. A final linear layer produces a scalar regression output (QM9, QM40, ESOL, LIPO, FreeSolv) or a logit for classification (BBBP, BACE), trained with mean-squared-error or binary cross-entropy loss respectively. We optimize with the AdamW optimizer for 200 epochs and a linear warmup scheduler. The final model is chosen by early stopping based on the validation loss. Regression targets are normalized to mean 0 and variance 1 before training, matching the GRALE protocol of (Krzakala et al., 2025, App. A.1), and reported MAE values are computed in the normalized label space. The five regression tasks are scored by normalized mean absolute error and the two classification tasks by ROC-AUC, following (Krzakala et al., 2025, Table 3); per-task means and standard deviations are reported in the main-paper Table 3 and reproduced with per-split standard deviations in Table G.10.

Fingerprint2Graph translator. The translator $\psi : \mathbb{R}^{2048} \to \mathbb{R}^{2D}$ maps a folded 2048-bit Morgan/ECFP4 fingerprint, of radius 2, computed via RDKit, to a target hypervector $\mathbf{h} = [\mathbf{h}_{\text{edge}} \,\|\, \mathbf{h}_{\text{graph}}] \in \mathbb{R}^{2D}$. Architecturally, ψ is a residual SwiGLU MLP organized in four stages, each consisting of a linear projection at width 2048 followed by two SwiGLU residual blocks at the same width. A final linear projection produces the $2D$-dimensional hypervector. The model is trained with a MSE reconstruction loss between $\psi(\text{ECFP4}(G))$ and the target hypervector. We optimize

with an AdamW optimizer for a maximum of 500 epochs, use a linear warmup and cosine decay scheduler, and determine the final model based on the best validation loss.

D.8 Generation Pipeline

We explore two architecture variations for the molecular generation task: GRAFT+FM trains a continuous flow matching model directly on the empirical distribution of the GRAFT dual stack representations of the ZINC250k dataset. GRAFT+AE+FM first trains an autoencoder that compresses the $2D$-dimensional hypervector into a smaller continuous latent, then trains a separate flow-matching model on that latent. Both variants share the same source distribution (standard Gaussian) and the same GRAFT decoder (frozen, applied at inference to map sampled hypervectors back to graphs). We follow the conditional flow-matching formulation of Lipman et al. (2023), with the addition of minibatch optimal-transport coupling between source and target samples (Tong et al., 2024).

Flow-matching velocity network. The velocity field $\mathbf{v}_\theta(\mathbf{x}_t, t)$ is parameterized by a residual MLP with SwiGLU + adaLN-Zero conditioning blocks. The diffusion time $t \in [0, 1]$ is encoded by a 128-dimensional sinusoidal embedding and injected through adaLN-Zero gating; sampling of t during training follows a logit-normal schedule. The probability path is the conditional-OT path $\mathbf{x}_t = t\mathbf{x}_1 + (1-t)\mathbf{x}_0$ with Gaussian source and minibatch-OT-coupled target. GRAFT+FM is trained with a standard MSE velocity-matching loss, while GRAFT+AE+FM uses a Gaussian-mixture negative-log-likelihood loss with $K = 8$ mixture components on the velocity prediction. Both variants are optimized with AdamW for 500 epochs and use an EMA shadow of the parameters at sampling time.

Hypervector autoencoder (GRAFT+AE+FM only). The autoencoder is a symmetric SwiGLU residual stack with a latent dimension of 256. The encoder is a linear projection from $2D$ into the trunk, followed by 8 SwiGLU residual blocks (FFN multiplier $8/3$, residual dropout 0.025) and a final linear projection into the 256-dimensional latent; the decoder mirrors this structure in reverse. The model is trained with an MSE reconstruction loss on the $2D$-dimensional hypervector, optimized with AdamW for 500 epochs with gradient clipping at 1.0; evaluation weights are taken from an EMA shadow of the parameters with decay 0.999.

Sampling. At inference time we draw $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and integrate the learned velocity field with a midpoint ODE solver for $1\,000$ steps to obtain a sampled hypervector. For GRAFT+AE+FM the resulting latent is first passed through the frozen AE decoder to recover the full $2D$-dimensional hypervector. The hypervector is then handed to the frozen GRAFT decoder, which produces a graph with node attributes; the bond-order post-processing of App. D.6 together with RDKit sanitization yields the final molecule.

D.9 Compute

Experiments across this paper were run on three different GPUs: an NVIDIA RTX 2060, an NVIDIA RTX 5060 Ti, and an NVIDIA H100.
The reconstruction runs of the main paper were executed partially on the RTX 5060 Ti and partially on an H100 GPU, depending on graph-size regime. Including the ablation sweeps of App. E and the hyperparameter exploration that selected the headline configuration, the reconstruction track required approximately 350 GPU-hours time across roughly 80 inference-only runs.
The translator MLPs ψ for the QM9 and GDB13 Fingerprint2Graph experiments were one-off models trained for approximately 24 hours each on an RTX 2060.
Due to the small dataset sizes of the MoleculeNet tasks, each MLP head trains in only a few minutes on an RTX 2060; total compute across the seven tasks and split repetitions is on the order of a few GPU-hours.
Hyperparameters of the flow-matching model and the autoencoder were tuned on an H100 with approximately 30 runs each (60 in total) at an average runtime of 6 hours per run, amounting to roughly 360 GPU-hours total.

D.10 Software environment and code availability

All experiments use Python 3.12, PyTorch 2.10, CUDA 12.8, PyTorch Lightning 2.6.1, PyTorch Geometric 2.7.0, torchhd 5.8.4, and NumPy 1.26.4, alongside the RDKit 2025.9.5.

Table E.4: Hypervector-dimension sweep at the ablation center on PubChem-32 ($n = 5\,000$ molecules, seed 42).

D	Edge acc. (%)	Graph acc. (%)	Re-encoded cosine
16	0.69	0.24	0.488
32	2.60	0.46	0.411
64	9.79	3.48	0.345
128	37.85	22.72	0.383
256	92.10	74.08	0.899
512	99.96	90.08	0.999
1024	99.96	90.46	0.999

The complete implementation, including the parameter-free codec, the matching-pursuit and beam-search decoders, the Fingerprint2Graph projection head, the property-prediction heads, and the flow-matching generation pipeline, will be released under a permissive open-source license upon publication.

E Ablation Studies

This section reports controlled sweeps over the components of the GRAFT codec. All single-axis sweeps are run around the *ablation center* $(D, L, \mathcal{K}, B, B_{\text{beam}}) = (1024, 3, \{6, 10, 14\}, 8, 32)$ on $5\,000$ PubChem-32 molecules with seed 42, deliberately chosen as a smaller and faster operating point than the headline configuration (App. D.2) so each axis can be swept densely.

E.1 Theoretical HRR capacity bound

The dimension panel of the main-paper ablation figure overlays the empirical curve with an information-theoretic ceiling that we derive here. The bound assumes i.i.d. Gaussian-random hypervectors (Plate, 1995; Kanerva, 2009), under which two random vectors in $\mathbb{R}^D$ have zero-mean cosine similarity with standard deviation $\approx 1/\sqrt{D}$.

For a graph with k edges, encoded as $\mathbf{h}_{\text{edge}} = \sum_{i=1}^{k} \mathbf{x}_{a_i} \circledast \mathbf{x}_{b_i}$, recovering a single edge term by similarity search against the codebook reduces to picking the correct entry against $k - 1$ noise cross-terms. Modeling those cross-terms as independent Gaussians with variance $1/D$, the correct entry wins with probability $\Phi\big(\sqrt{D/(k-1)}\big)$, where Φ is the standard normal CDF. Exact graph recovery requires all k edges to win their argmax, which under independence gives

$$P_{\text{graph}}(D, k) \;\approx\; \Phi\left(\sqrt{\tfrac{D}{k-1}}\right)^{k}. \tag{E.3}$$

Plugging in $k = 19$ — the average heavy-atom and edge count of a PubChem-32 graph — yields the dashed sky-blue curve in the main-paper ablation figure.

The bound is decoder-unaware in two ways. First, it treats each edge recovery as a one-shot query against the original $\mathbf{h}_{\text{edge}}$, whereas GRAFT subtracts each recovered term from a running residual — so any single-step error compounds into subsequent steps. Second, it has the argmax beat only random distractors, whereas the actual codebook contains every observed atom-pair binding ($\sim |\mathcal{X}^\star|^2$ candidates) and the search must rank against thousands of near-orthogonal alternatives. Both effects push the empirical curve below the bound; the gap visible in the main-paper figure is the cost of the realistic greedy decoder relative to this oracle ceiling.

E.2 Hypervector dimension D

Graph accuracy follows the HRR capacity bound (Sect. E.1) closely, plateauing just below the theoretical ceiling (Fig. E.4a). Edge accuracy is near-perfect by $D = 512$, while graph accuracy plateaus at $\sim 90\%$ for $D \geq 512$. Doubling further to $D = 1024$ only adds 0.4 percentage points — the remaining error is no longer about capacity but about the topology beam search. We use $D = 1024$ as a conservative buffer that also leaves headroom for the larger graphs of PubChem-64.

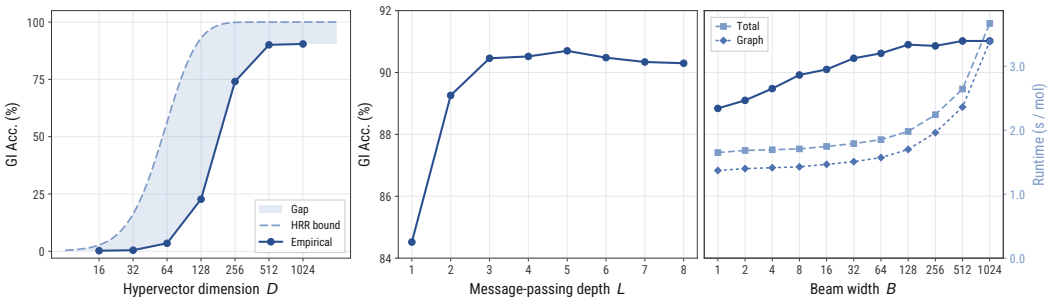


Figure E.4: Architectural ablation at the ablation center on PubChem-32. (a) GI Acc. (navy) vs. hypervector dimension D, overlaid on the HRR capacity bound (sky dashed, Eq. E.3 at $k = 19$); the shaded region between the empirical curve and the bound is highlighted. (b) GI Acc. vs. message-passing depth L. (c) GI Acc. (left axis, navy) and runtime (right axis: total, sky dashed; graph decode, azure dashed) vs. beam width B_{beam}.

Table E.5: RW-bin sweep at the ablation center on PubChem-32. $|\mathcal{X}^\star|$ is the realized node-feature vocabulary; the edge-codebook column is the count of observed atom-pair bindings $|\mathcal{E}^\star|$ stored at $D = 1024$.

B	$\lvert\mathcal{X}^\star\rvert$	Edge codebook	Graph acc. (%)
4	1 502	49 306	84.90
6	2 539	96 399	88.76
8	3 644	158 636	90.46
10	5 118	231 094	90.82
12	6 898	327 446	91.14

E.3 Message-passing depth L

Graph accuracy climbs steeply from $L = 1$, plateaus by $L = 3$, peaks at $L = 5$, and decreases slightly for $L > 5$ (Fig. E.4b). The plateau at $L = 3$ matches the encoder's 1-WL ceiling for the average PubChem-32 graph (mean atom count 19). We use $L = 5$ as the headline value because it gives additional receptive-field margin for the larger graphs of PubChem-64.

E.4 Beam width B_{beam}

Graph accuracy gains roughly 2 pp across a $1024\times$ beam sweep, with most of the gain at small B_{beam} and the curve plateauing by $B_{\text{beam}} = 512$ (Fig. E.4c). Decoding wall-clock grows sub-linearly: the same $1024\times$ widening adds only $\sim 2.5\times$ per-graph time. We use $B_{\text{beam}} = 32$ in the headline configuration as a balance between accuracy and decoding time.

The plateau itself is informative: wider beams keep recovering correct topologies that narrow beams had pruned, but each doubling also surfaces partial assemblies that re-encode to the *same* hypervector as the target — confusions that no $\mathbf{h}_{\text{graph}}$-based scorer can distinguish. Pushing past the plateau therefore requires encoder-side changes (richer node features or deeper L) rather than wider search.

E.5 RW bins B

Graph accuracy climbs steeply from $B = 4$ to $B = 8$ and plateaus thereafter (Table E.5, Fig. E.5a). The bin count is the ablation axis that most directly drives edge-decoder cost: more bins enlarge the edge codebook, and the per-step similarity search scales with its size. We use $B = 8$ in the headline configuration as the operating point past which further bins inflate decoding time with only small accuracy gains.

Table E.6: Walk-length-set sweep at the ablation center on PubChem-32, group means over the 33 cells of Fig. E.5b.

$\lvert\mathcal{K}\rvert$	Cells	Mean graph acc. (%)	Mean edge acc. (%)
1	3	83.88	99.96
2	9	88.78	99.96
3	9	90.27	99.96
4	12	90.59	99.96

E.6 RW WALK-LENGTH SET $\mathcal{K}$

The walk-length set has two distinct effects on the encoder. First, increasing the cardinality $|\mathcal{K}|$ adds another column of bin indices to the node feature tuple, which expands the realized vocabulary $|\mathcal{X}^\star|$. Second, the choice of arithmetic step s and start length k_0 at fixed $|\mathcal{K}|$ controls which scales of local topology the encoder probes through the structural pattern of $\mathbf{M}^k$.

The two effects behave very differently. Cardinality is the dominant lever: moving from no random-walk descriptor at all to a single walk length already lifts graph accuracy by tens of percentage points, while extending from one to four walk lengths adds only a small fraction of that initial jump (Table E.6). The diminishing returns are unsurprising — $\mathbf{M}^k$ for nearby k probes the same local structure, so each additional walk length distinguishes a smaller fraction of node neighborhoods than the previous one. By contrast, s and k_0 at fixed $|\mathcal{K}|$ shift accuracy by far less (Fig. E.5b): the encoder is largely indifferent to which scales are sampled, as long as enough are sampled to grow the vocabulary.

The reason cardinality matters is the realized vocabulary $|\mathcal{X}^\star|$ that grows with it (App. E.7). Edge accuracy stays unchanged across the sweep, while graph accuracy falls by several percentage points at $|\mathcal{K}| = 1$. The two decoding stages respond to vocabulary size differently. Edge recovery only has to identify which edge types are present, and that step works fine with a smaller vocabulary — it just sees fewer types. The beam search, by contrast, has to decide which specific atom each node represents, and a smaller vocabulary leaves more atoms looking interchangeable, inflating the candidate space at every search step.

E.7 VOCABULARY PRESSURE AS THE UNIFYING LEVER

The graph decoder is an enumerator over candidate graphs, and the realized node-feature vocabulary $|\mathcal{X}^\star|$ controls how ambiguous each candidate is: a larger vocabulary partitions atoms into more distinct types, so each anchor produces fewer interchangeable extensions per step. Both the bin count B and the walk-length cardinality $|\mathcal{K}|$ affect graph accuracy through this vocabulary, so their effects on accuracy largely overlap.

The empirical sweeps confirm this. Across the $n = 41$ cells of the combined bins and $\mathcal{K}$ sweeps, Spearman's rank correlation between $|\mathcal{X}^\star|$ and graph accuracy is $\rho = +0.95$ (Fig. E.5c). The relationship saturates: graph accuracy rises steeply below $|\mathcal{X}^\star| \approx 3\,000$ and is nearly flat above $|\mathcal{X}^\star| \approx 5\,000$, where doubling the vocabulary adds at most one percentage point. The headline configuration sits past saturation by design, so the small $|\mathcal{X}^\star|$ shifts that come from changing the heavy-atom cap (16 to 32 to 64) move the operating point along the flat region rather than the steep one — the codec stays robust across sizes without per-cap retuning.

E.8 EMPIRICAL DECODING-TIME COMPLEXITY

Per-graph decoding wall-clock empirically scales as approximately $\mathcal{O}(n^2)$ on small graphs (PubChem-16) and between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ on the larger PubChem-32 and PubChem-64 sets (Fig. E.6). On PubChem-64 the variance across molecules above $n \approx 40$ is large enough that the exact exponent is hard to determine. This is consistent with the decoder transitioning from full enumeration to bounded beam search at larger sizes (App. D.5): per-molecule cost then depends on how often the beam has to be pruned.

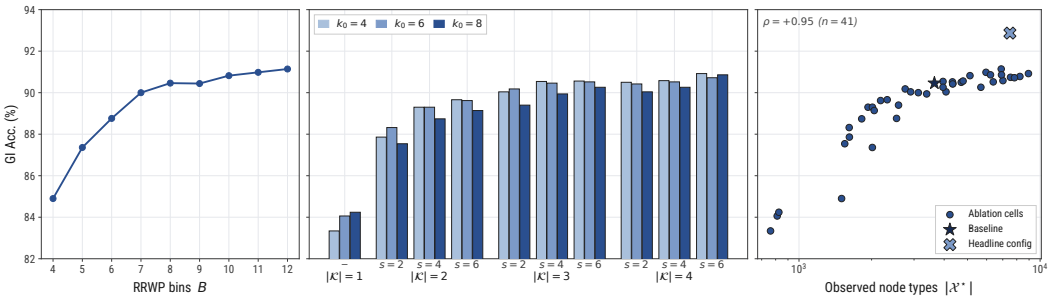


Figure E.5: RW-discretization ablation at the ablation center on PubChem-32. (a) GI Acc. vs. bin count B. (b) GI Acc. by walk-length set $\mathcal{K}$, grouped by cardinality $|\mathcal{K}|$ (outer block), arithmetic step s (sub-cluster), and start length k_0 (light-to-dark inner bars). (c) Cells of (a) and (b) plotted against the realized node-feature vocabulary $|\mathcal{X}^\star|$ on a log axis; the ablation center baseline (black star) and the headline configuration (sky $\times$) are overlaid, and the annotation reports Spearman ρ across cells.

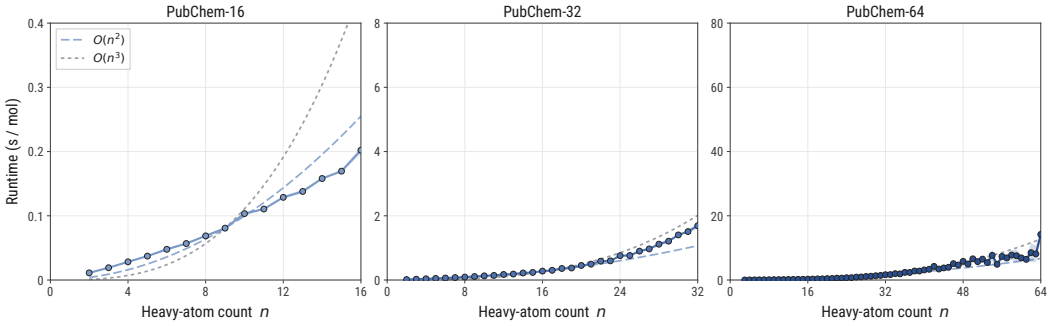


Figure E.6: Decoding wall-clock per molecule vs. heavy-atom count n at the headline configuration, one panel per dataset (PubChem-16 / 32 / 64). Lines are the per-n mean over three seeds at $n_{\text{test}} = 10\,000$ molecules per seed; bands are sample standard deviation. Reference curves $\mathcal{O}(n^2)$ (sky dashed) and $\mathcal{O}(n^3)$ (gray dashed) are anchored at the median n of each dataset.

F Reconstruction Failure-Mode Breakdown

Reconstruction failures fall into three categories. Edge-decoding failure (f1) arises when matching pursuit returns an incorrect edge multiset, so the decoded graph has the wrong set of edges. Encoder-isomorphism failure (f2a) is when the decoder returns a graph $\hat{G} \ncong G$ that re-encodes to the same hypervector as the input G: the encoder cannot tell the two apart, so no decoder choice can recover the correct graph without changing the encoder itself. Beam-search failure (f2b) is when $\hat{G}$ is incorrect and re-encodes to a different hypervector than the input: the bounded search has not explored a candidate that re-encodes to the same hypervector as G, but a wider beam could in principle have reached one.

With $D \geq 1024$, edge-decoding failure is essentially absent ($< 0.4\%$ of failures on any dataset, $\sim 0.3\%$ pooled): the recovered edge multiset matches the input on $\geq 99.99\%$ of attempts, and the rest of the error sits in the beam-search reconstruction. The remaining failures are dominated by encoder isomorphism: $\sim 74\%$ of failures pooled and at least 68% on every individual variant, with beam-search failures making up the remaining $\sim 26\%$. This proportion is broadly stable across the PubChem variants despite a $\sim 7\times$ range in absolute failure rate (Table F.7).

Encoder-isomorphic failures (f2a). f2a is the consequence of the encoder's 1-WL expressiveness ceiling: when 1-WL cannot distinguish two non-isomorphic graphs, neither can the encoder, and the decoder has no signal to prefer one over the other.

Table F.7: Failure-mode breakdown with $D \geq 1024$ (474 000 decoded molecules across 98 configurations). Shares are conditional on reconstruction failure. The pooled row covers three seeds at $n_{\text{test}} = 10\,000$ molecules per seed.

Scope	Failure rate	f1 share	f2a share	f2b share
Pooled (all datasets)	9.23%	0.31%	73.74%	25.95%
PubChem-16	1.49%	0.27%	68.10%	31.64%
PubChem-32	10.09%	0.36%	74.58%	25.07%
PubChem-64	10.38%	0.11%	70.95%	28.94%

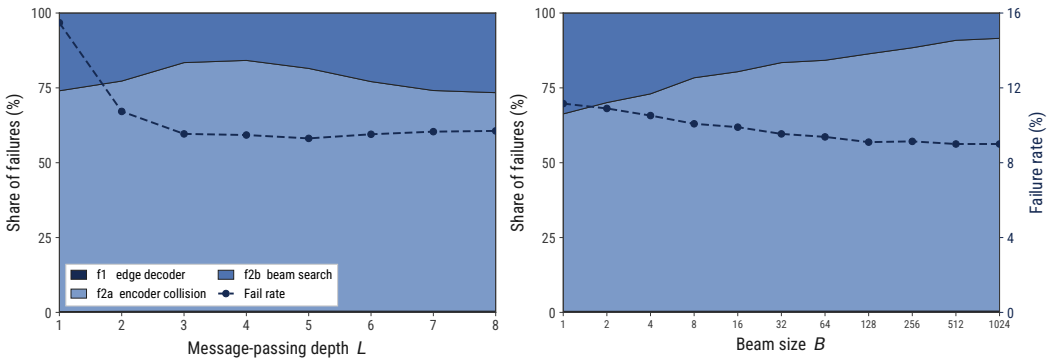


Figure F.7: Absolute failure rates per mode as a function of message-passing depth L (left) and beam width B_{beam} (right) at the ablation center on PubChem-32 ($n = 5\,000$ molecules, seed 42).

Depth L has opposite effects on the two failure modes (Fig. F.7, left): f2a decreases monotonically with depth (a 38% relative reduction from $L = 1$ to $L = 8$), while f2b traces a U with a minimum near $L = 4$. The U arises because at low L the per-node hypervectors are too local for the beam scorer to distinguish near-tied candidates, while at high L message passing has already converged to its 1-WL ceiling and additional rounds dilute earlier-layer signal through over-bundling. The headline $L = 5$ sits at the total-failure minimum where these two effects balance.

Beam-search failures (f2b). The dominant lever for f2b is the search budget: widening the beam from $B_{\text{beam}} = 1$ to $B_{\text{beam}} = 1024$ reduces f2b roughly $5\times$, and the curve has not yet plateaued at the widest setting tested (App. E.4, Fig. F.7 right). A second, upstream lever is the realized vocabulary $|\mathcal{X}^\star|$: a larger vocabulary shrinks the per-anchor candidate space, so the search reaches a given depth before triggering pruning. The empirical correlation between $|\mathcal{X}^\star|$ and graph accuracy in App. E.7 therefore mostly reflects f2b reductions rather than f2a.

Wider beams shift the residual bottleneck from search to encoder: f2b's relative share among failures drops from $\sim 34\%$ at $B_{\text{beam}} = 1$ to $\sim 8\%$ at $B_{\text{beam}} = 1024$, while f2a stays nearly flat in absolute terms. A small fraction of the additional cases now classified as f2a are partial assemblies that narrower beams had pruned before re-encoding could recognize them (~ 0.84 pp at $B_{\text{beam}} = 1024$), consistent with f2a and f2b being largely independent failure mechanisms.

G Additional Comparisons to SOTA Graph Autoencoders

This section reports comparisons between GRAFT and learned graph-level autoencoders along axes that do not fit cleanly into the accuracy-focused tables of the main paper. Sect. G.1 compares compute budgets and discusses how those budgets amortize across the dataset sizes typical of pretraining and downstream evaluation.

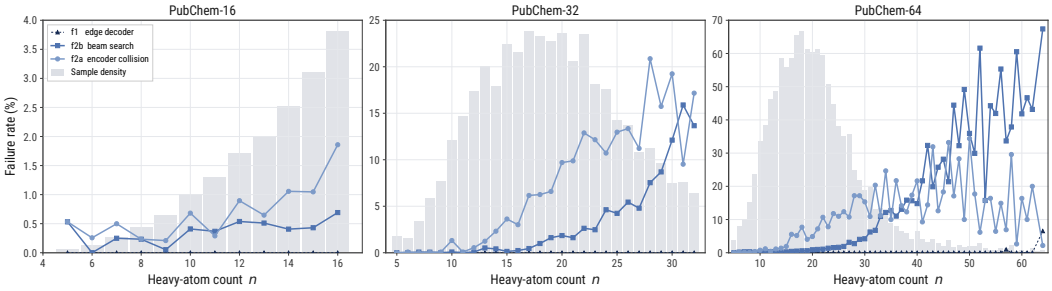


Figure F.8: Per-mode absolute failure rates as a function of heavy-atom count n, on PubChem-16 (left), PubChem-32 (center), and PubChem-64 (right). With $D \geq 1024$, pooled over configurations within each dataset.

Table G.8: Per-sample decoding compute as a function of graph size N. GRALE rows are taken from (Krzakala et al., 2025, App. C, Table 12) as training-step (forward + backward) values at batch 1 on an RTX 3500 Ada (12 GB). GRAFT decode times are means over three seeds at the headline configuration ($D = 1024$, $L = 5$, $B_{\text{beam}} = 32$), measured on the H100 cluster on $n_{\text{total}} = 402, 1601, 361, 19$ samples at $N = 8, 16, 32, 64$. The Memory column is the per-graph GPU runtime peak measured locally on an RTX 5060 Ti; the resident edge codebook is a one-time cost shared across all decoded graphs and is not counted per graph. † Extrapolated from a power-law fit on $N \in \{16, 32, 64\}$; not directly measured.

Model	Quantity	N=8	N=16	N=32	N=64	N=128	N=256
GRALE	Memory (GB)	0.2	0.3	0.4	0.8	3.0	10.7
GRALE	Step time (s)	0.01	0.01	0.02	0.02	0.03	0.07
GRAFT	Memory (GB)	0.43	0.43	0.52	0.55	$\approx 0.63^{\dagger}$	—
GRAFT	Decode time (s)	0.05	0.29	2.85	81.38	$\approx 1\,135^{\dagger}$	—

G.1 Compute Budget

The reconstruction comparison in Sect. 5.1 of the main paper measures accuracy and abstracts away the compute differences between GRAFT and the learned baselines. Those differences are real and asymmetric — and worth surfacing for the reader who wants to weigh the methods on a level playing field.

Memory and per-sample decoding cost. Learned graph-level autoencoders pay most of their compute at training time, but inference still scales steeply: GRALE's Evoformer-based architecture maintains an explicit pair representation that grows as $\mathcal{O}(N^3)$ in memory and wall-clock (Krzakala et al., 2025), both at training and at decoding time. This cubic memory cost is what caps the largest GRALE-trainable graph at $N \approx 32$ on commodity GPUs. In contrast, GRAFT's decoder scales differently: the encoder is independent of N in memory, and the decoder's per-graph working set is dominated by a fixed-size beam of partial reconstructions and a per-step similarity matrix multiplication against the edge codebook. The codebooks at the headline configuration (≈ 0.46 GB on PubChem-64: ≈ 0.43 GB edge codebook + ≈ 0.03 GB node codebook at $D = 1024$, float64) are a one-time GPU-resident cost loaded once per dataset and shared across all decoded graphs, so we do not count them in the per-graph column. The remaining per-graph GPU memory is roughly flat in N: ≈ 0.43 GB across $N \leq 16$, growing only mildly to ≈ 0.52 GB at $N = 32$ and ≈ 0.55 GB at $N = 64$. Table G.8 reports the per-sample numbers; the GRALE rows are taken from (Krzakala et al., 2025, App. C, Table 12) as training-step values, which we treat as upper bounds on inference since (Krzakala et al., 2025) does not report inference-only numbers.

GRALE's memory budget grows steeply in N and hits a hard ceiling on commodity GPUs, while GRAFT maintains a roughly N-flat per-graph footprint at well under 1 GB. The trade-off is a larger absolute decoding wall-clock, but no hard memory ceiling.

Table G.9: Training cost and amortized total compute per decoded sample on PubChem32. Training-hour figures are taken from (Krzakala et al., 2025, Table 10) and converted to GPU-hours using the GPU count reported there (1–2× L40S); we did not rerun any of the baselines ourselves. The per-sample decoding column for the learned baselines uses GRALE's published step time at N=32 as a representative value (Table G.8). The amortized columns are computed as $T_{\text{train}}/n + T_{\text{decode}}$ for $n \in \{10^4, 10^5, 10^6\}$.

Model	Train (GPU-h)	Decode (s)	Amort. n=10^4 (s)	Amort. n=10^5 (s)	Amort. n=10^6 (s)
PIGVAE (2021)	95	0.02	34.20	3.44	0.36
GRALE (2025)	200	0.02	72.02	7.22	0.74
GRAFT	**0**	2.85	2.85	2.85	2.85

Training cost and amortization. The comparison above is inference-only. A full-pipeline comparison must also account for the one-time training cost of the learned baselines — which GRAFT does not pay, as the codec is training-free and the only setup cost is constructing the random codebook (negligible relative to any of the training budgets in Table G.9). Table G.9 reports the published PubChem32 training cost of PIGVAE and GRALE (Krzakala et al., 2025, Table 10), together with the per-sample compute that results from amortizing each training run over the dataset on which the trained checkpoint is eventually used. We consider three dataset scales: $\sim 10^4$, $\sim 10^5$, and $\sim 10^6$ samples.

Including the one-time training cost changes the comparison substantially. At $\sim 10^4$ samples — the scale of a typical test set or downstream evaluation — almost all of GRALE's per-sample compute comes from amortizing its training run, and GRAFT comes out ahead on cumulative compute despite its slower decoder. The two costs balance at roughly 2.5×10^5 samples decoded with the same trained checkpoint, and beyond that GRALE's faster per-sample step wins on net.
While GRAFT is not uniformly cheaper than GRALE — at any fixed N, GRALE's per-sample decoding wall-clock is lower in absolute terms — we see the comparison between a training-free and a trained codec as inherently two-sided. The relative ordering depends on how many graphs the trained checkpoint will eventually decode, and a complete accounting must therefore include the one-time training cost that GRAFT avoids entirely.

G.2 Property Prediction

The main-paper property-prediction table (Table 3 in Sect. 5.3 of the main paper) reports means only. For completeness we reproduce the same table here with the per-split standard deviations. Baseline means and standard deviations are taken directly from (Krzakala et al., 2025, Table 3); GRAFT numbers are means±std over 5 train/test splits using the unified-MLP head of Sect. 5.3 of the main paper.

Table G.10: MoleculeNet property prediction with standard deviations. Regression metric is normalized MAE (↓); classification metric is ROC-AUC (↑). Baseline numbers are taken from (Krzakala et al., 2025, Table 3); GRAFT numbers are means±std over 5 train/test splits. Best mean per column in **bold**.

	Regr. (norm. MAE ↓)					Class. (ROC-AUC ↑)	
Model	QM9 (U_0)	QM40 (U_0)	ESOL	LIPO	FreeSolv	BBBP	BACE
SimGRACE (2022)	0.110 ±0.003	0.025 ±0.005	0.293 ±0.020	0.534 ±0.023	0.374 ±0.008	0.745 ±0.072	0.866 ±0.050
InfoGraph (2020)	0.122 ±0.001	0.020 ±0.001	**0.255** ±0.016	0.495 ±0.013	0.297 ±0.010	0.729 ±0.053	0.845 ±0.046
GraphMAE (2022)	0.222 ±0.016	0.247 ±0.036	0.291 ±0.012	0.527 ±0.029	0.378 ±0.028	0.773 ±0.036	0.857 ±0.039
GVAE (2016)	0.765 ±0.005	0.328 ±0.005	0.306 ±0.027	0.668 ±0.014	0.344 ±0.022	0.705 ±0.060	0.771 ±0.049
PIGVAE (2021)	0.031 ±0.001	0.019 ±0.001	0.279 ±0.020	0.523 ±0.019	0.283 ±0.013	0.675 ±0.079	0.816 ±0.049
GRALE (2025)	0.015 ±0.001	0.018 ±0.003	0.274 ±0.014	0.511 ±0.022	**0.272** ±0.017	0.731 ±0.025	0.821 ±0.051
GRAFT	0.119 ±0.002	0.196 ±0.002	0.472 ±0.026	0.563 ±0.008	0.574 ±0.074	0.848 ±0.053	0.846 ±0.073
GRAFT (w/o RW)	**0.011** ±0.0001	**0.0075** ±0.0002	0.289 ±0.019	**0.451** ±0.008	0.310 ±0.056	**0.857** ±0.060	**0.860** ±0.051