FRAGLINGO: OPEN-VOCABULARY MOLECULAR DESIGN VIA ATTACHMENT-AWARE FRAGMENT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce **Fraglingo**, an autoregressive molecular generator that constructs molecules step by step from *fragments* — chemically meaningful substructures connected through predefined attachment sites. At each generation step, the current partial molecule is the *growing molecule*, and one of its available attachment sites is selected as the *active attachment site* from which generation continues. Fraglingo jointly predicts what fragment to add and how it should attach by predicting an *attachment-aware fragment embedding*, a continuous representation that encodes both fragment identity and attachment configuration, and retrieving the nearest fragment through latent-space search. To capture attachment context, we introduce a *wildcard-anchored readout* that represents both the growing molecule and candidate fragments relative to their attachment sites, enabling a single latent prediction to determine both which fragment to attach and where the new bond should form. Because prediction operates in a continuous embedding space rather than over fixed fragment identifiers, larger fragment libraries can be introduced at inference time without retraining. This retrieval-based formulation provides a unified generation primitive for molecule generation, scaffold generation, scaffold decoration, and molecule optimization. Fraglingo also supports *property-conditional generation*, where desired molecular properties are provided as generation conditions to control the properties of generated molecules. On controlled property-conditional benchmarks, Fraglingo achieves stronger joint property control than comparably trained baselines while maintaining competitive validity, uniqueness, and novelty, and generalizes to fragment libraries up to $4\times$ larger than those used during training without retraining. *Code is available at:* `https://anonymous.4open.science/r/FragLingo-3551`.

1 INTRODUCTION

Molecular design aims to discover chemical structures with desired biological and physicochemical properties, and is a central problem in drug discovery. Generative models provide a way to search this enormous chemical space by learning how to construct new molecules while satisfying objectives such as potency, drug-likeness, or similarity to a known scaffold. A fundamental design choice is therefore the *unit of generation*: what does the model predict at each step? Existing approaches generate molecules as sequences of SMILES (Weininger, 1988) or SELFIES (Krenn et al., 2020) tokens, atoms and bonds in molecular graphs, functional modules (Edwards et al., 2026), or entire molecular graphs through diffusion and flow models (Segler et al., 2018; Gómez-Bombarelli et al., 2018; Ross et al., 2022; Shi et al., 2020; Hoogeboom et al., 2022). These representations are expressive, but their generation units are often lower-level than the structural changes considered during molecular design. For example, changing or misplacing a single parenthesis or ring-closure token in SMILES can alter the molecular connectivity or make the string invalid, while atom-by-atom graph generation may require many sequential decisions to construct a single recognizable chemical motif.

Fragment-based generation instead constructs molecules from larger, chemically meaningful substructures. This more closely resembles common medicinal-chemistry operations such as adding, replacing, extending, or decorating molecular motifs (Hajduk & Greer, 2007; Kirsch et al., 2019). A molecule is first decomposed into *fragments*, with the broken bonds represented as *attachment*

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.

Table 1: Key terminology used in Fraglingo.

Term	Definition
Fragment	A chemically meaningful molecular substructure used as a generation unit.
Attachment site	A position on a fragment where a bond to another fragment can be formed, represented by a wildcard atom (*).
Active attachment site	The open attachment site of the growing molecule from which the next generation step proceeds.
Growing molecule	The partially assembled molecule at the current generation step.
Attachment-aware	A representation conditioned on the specific attachment site through which a fragment participates in assembly.
Wildcard-anchored readout	A graph readout that represents a molecule relative to a selected wildcard attachment site.

sites. Generation then proceeds autoregressively: given a partially constructed *growing molecule*, the model selects an open attachment site and predicts a fragment to connect there. Table 1 summarizes the terminology used throughout this work.

Despite its advantages, fragment-based generation introduces two fundamental challenges. First, many existing approaches formulate next-fragment prediction as classification over a fixed vocabulary (Jin et al., 2020; Kong et al., 2022; Yue et al., 2024). The model can therefore directly select only fragments represented in its training-time output space, making it difficult to exploit larger or updated fragment libraries at inference time. Second, predicting *which* fragment to use is not sufficient: the model must also determine *how* it attaches. A growing molecule may contain several open attachment sites, and a candidate fragment may itself have multiple possible attachment points. As illustrated in Figure 1, the same pair of fragments can therefore produce different molecular structures depending on which attachment sites are connected. A global representation of either molecule does not explicitly distinguish these local attachment configurations.

We introduce **Fraglingo**, an attachment-aware autoregressive fragment generator designed to address both challenges. Fraglingo decomposes molecules into fragments and repeatedly predicts the next fragment to attach to the growing molecule. The framework is agnostic to the fragmentation scheme; we use BRICS (Degen et al., 2008) as the primary scheme and BFE (Nguyen & Ji, 2026) in an ablation study. Rather than predicting a discrete fragment identifier, Fraglingo performs *continuous latent retrieval*: the model predicts an embedding for the desired next fragment and retrieves its nearest neighbor from a fragment embedding library. The output space is therefore defined by learned chemical representations rather than fixed class identifiers, allowing the retrieval library to be expanded at inference time without retraining.

The key challenge is making this retrieval *attachment-aware*. Fraglingo introduces a *wildcard-anchored readout* that constructs representations relative to a selected attachment site. The growing molecule is encoded from the perspective of its active attachment site, and candidate fragments are represented from the perspective of each possible attachment site. Retrieval therefore asks a local and chemically explicit question: *which fragment, viewed through which attachment site, best continues this particular site of the growing molecule?* A single retrieved representation consequently determines both the fragment identity and its attachment configuration. Desired molecular properties can additionally be supplied as conditions, allowing the same generation process to steer molecules toward specified property profiles.

This attachment-aware retrieval primitive supports four molecular design tasks within the same framework: molecule generation, scaffold generation, scaffold decoration, and molecule optimization. Molecules and scaffolds are constructed through autoregressive fragment assembly, scaffolds are decorated by retrieving branches at specified attachment sites, and existing molecules are optimized through conditioned fragment replacement. Across these settings, Fraglingo retains the same basic operation: predict an attachment-aware representation and retrieve a fragment from a continuous embedding space.

In summary, our contributions are:

- **Wildcard-anchored readout for attachment-aware generation.** We introduce a site-specific graph readout that represents both the growing molecule and candidate fragments relative to their attachment sites, enabling joint prediction of *what* fragment to add and *where* to attach it.

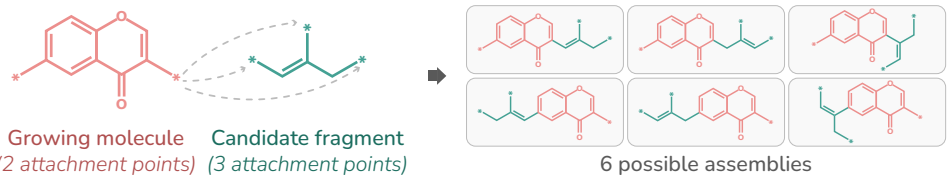


Figure 1: Attachment ambiguity in fragment-based molecular generation. The same pair of fragments can produce different molecular structures depending on which attachment sites are connected.

- **Continuous latent retrieval for open-vocabulary fragment generation.** We formulate next-fragment prediction as retrieval in a learned embedding space rather than classification over fixed fragment identifiers, enabling the fragment library to be expanded at inference time without retraining.
- **One retrieval primitive for four molecular design tasks.** Fraglingo supports molecule generation, scaffold generation, scaffold decoration, and molecule optimization within a unified fragment-retrieval framework.

2 Molecular Design via Latent-Space Autoregressive Fragment Generation

This section first introduces the common generation primitive used by Fraglingo, then explains how the same primitive is adapted to molecule generation, molecule optimization, scaffold generation, and scaffold decoration. Detailed model design, training objectives, and configuration choices are provided in Appendix E.

2.1 Molecule / Scaffold Generation

Fragment assembly as next-fragment prediction. Fraglingo generates a molecule by sequentially assembling chemically meaningful fragments. We first decompose each training molecule using BRICS, which breaks selected bonds and replaces them with wildcard attachment atoms. This produces a fragment tree $\{F_1, \ldots, F_T\}$. Starting from a cap fragment, assembly proceeds in breadth-first order. At step t, the model observes a partially assembled molecule G_t and an active wildcard w_t, and must predict the next fragment F_{t+1} and the wildcard through which it should attach. Scaffold generation follows exactly the same process, with Murcko scaffolds (Bemis & Murcko, 1996) used as the training structures instead of complete molecules. Figure 2 illustrates this generation process.

Why attachment-aware representations? A fragment cannot be represented only by its molecular identity because the same fragment may contain multiple attachment sites that lead to different assemblies. Fraglingo therefore represents both the growing molecule and candidate fragments from the perspective of a specific wildcard. A shared graph encoder ϕ, implemented as a 6-layer FragmentGPS network based on GPS (Rampášek et al., 2022), first computes atom representations. The designated wildcard then acts as an attention query over the real atoms of the graph, producing the *wildcard-anchored readout*: $e_1 = \phi(G_t, w_t)$ and $e_2 = \phi(F_{t+1}, w_{t+1})$, where e_1 represents the growing molecule from the active attachment site and e_2 represents the candidate fragment from the site through which it would attach. Consequently, a fragment with multiple wildcards has a different representation for each possible attachment site. This allows the prediction to specify both *which fragment* to retrieve and *where* that fragment should attach.

Why continuous fragment retrieval? Treating next-fragment prediction as classification restricts inference to a fixed set of fragment classes. Fraglingo instead predicts a point in the learned fragment embedding space. Desired molecular properties $\mathbf{p}$ are first incorporated into the attachment-aware context, $e_1^{\text{cond}} = e_1 + W_p\mathbf{p} + \mathbf{b}_p$, where we condition on logP, MW, QED, TPSA, HBD, HBA, and RotBonds computed with RDKit (Landrum, 2006). A two-layer MLP then maps this context to the predicted next-fragment embedding: $\hat{z} = f(e_1^{\text{cond}})$. Property conditions are randomly masked with probability 0.2 during training to enable classifier-free guidance at inference.

Learning the retrieval space. Training must make $\hat{z}$ close to the correct attachment-aware fragment embedding while maintaining a structured and discriminative fragment space. We jointly optimize

$$\mathcal{L} = \lambda_{\text{NCE}}\mathcal{L}_{\text{InfoNCE}}(\hat{z}, e_2) + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}(e_2) + \lambda_{\text{commit}}\mathcal{L}_{\text{commit}}, \tag{1}$$

where InfoNCE (Oord et al., 2018) aligns the prediction with the correct fragment embedding, the classification objective prevents the fragment space from collapsing, and the commitment loss regularizes the predicted embedding using a residual vector quantizer (RVQ). The RVQ contains four stages with 64 codes per stage and is used only during training. Grid search selects $\lambda_{\text{NCE}} : \lambda_{\text{cls}} : \lambda_{\text{commit}} = 1 : 0.5 : 0.25$.

Open-vocabulary inference. After training, both the classification head and RVQ are discarded. We encode every candidate fragment with the trained graph encoder and construct an attachment-aware embedding lookup table. At each generation step, Fraglingo predicts $\hat{z}$ and retrieves its nearest fragment embedding using cosine similarity; the corresponding fragment is then connected to the active wildcard. Because fragment selection depends on embeddings produced by the encoder rather than fixed output classes, new fragments can be encoded and added to the retrieval library after training without retraining the generator. Generation repeats until no open attachment sites remain.

2.2 Molecule Optimization

Molecule optimization asks for local edits that improve target properties while preserving much of the starting molecule. To condition generation on the molecule being optimized, we add a global reference encoder. This encoder summarizes the starting molecule so that the model can choose edits that are compatible with the original structure. Since a complete reference molecule has no open attachment point, we replace the wildcard-anchored readout with a learned-query attention pool (GlobalPool): a trainable query vector attends over all atoms of the reference molecule, yielding a global embedding e_{ref}. The predicted next-fragment embedding becomes: $\hat{z} = f(e_1 + W_p\mathbf{p} + \mathbf{b}_p + W_r e_{\text{ref}})$, so generation is jointly conditioned on the current partial assembly, target properties, and the reference

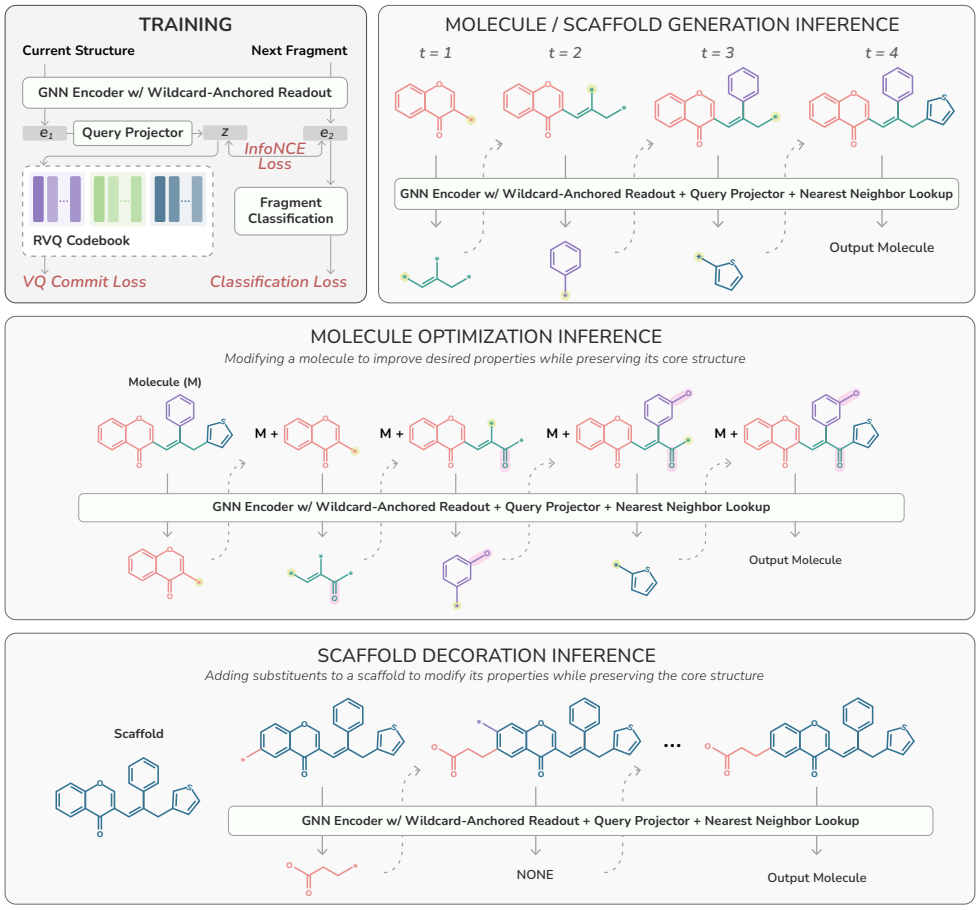


Figure 2: Overview of Fraglingo training and inference across four molecular design tasks. The architecture shown illustrates unconditional generation, with the GNN encoder and query projector abbreviated for clarity; detailed architectures are provided in Figure 8.

structure. During training, $\mathbf{p}$ and e_{ref} are each independently dropped out: dropping e_{ref} recovers plain conditional generation, while dropping $\mathbf{p}$ recovers reference-anchored generation without a property target. The optimized molecule is generated with the same fragment-retrieval process as molecule generation. Substitutions, insertions, and deletions all emerge naturally through the sequence of fragment predictions. For example, a deletion corresponds to generating a fragment tree with fewer nodes than the reference molecule. Similarity to the original molecule is not imposed by a hand-coded constraint; instead, the model learns the expected degree of preservation from paired training examples through the reference-conditioning signal.

2.3 Scaffold Decoration

Scaffold decoration uses the same retrieval primitive in a different setting. Given a fixed Murcko scaffold (Bemis & Murcko, 1996), the task is to independently predict which branch fragment should attach at each growable site. Unlike autoregressive generation, decoration is a per-site retrieval problem; the scaffold never changes, and each site is filled in a single forward pass. The only architectural change is the readout query. Instead of treating a wildcard as the active attachment site, the query is centered on the scaffold atom at the growable site. The attachment-anchored readout (Eq. 3) lets this atom attend to the full scaffold context, producing a representation that asks *which branch best fits this attachment site in this scaffold*. Candidate branches are drawn from a vocabulary of $\star$-R fragments, where $\star$ denotes the attachment atom, together with a learned `[NONE]` token that leaves the site undecorated. Property conditioning and the $\mathrm{MLP}_{\mathrm{Mapper}}$ predictor f are identical to those used for molecule generation. At inference, $\hat{z} = f(e_1^{\mathrm{cond}})$ is predicted independently for each growable site from the fixed, undecorated scaffold, and the branch whose embedding is nearest to $\hat{z}$ is retrieved from the full branch vocabulary by cosine similarity.

3 Experiments

3.1 Training Data

We construct task-specific grids of datasets to study how Fraglingo scales with both vocabulary size and corpus size. Molecules are first decomposed into fragments using the BRICS algorithm (Degen et al., 2008), and the fragment vocabulary is defined as the k most frequent BRICS fragments in the training corpus. Each grid is formed by choosing a vocabulary size k, retaining examples whose constituent fragments are fully covered by that vocabulary, and then subsampling a corpus of size N. The resulting $k \times N$ design lets us separate the effect of a larger retrieval library from the effect of more training examples while keeping the data-construction protocol consistent across tasks. All datasets derive from a shared base corpus of 883M commercially available molecules from ZINC15 (Sterling & Irwin, 2015), filtered to molecular weight below 500 Da and retaining only molecules that decompose into at least two BRICS fragments. Task-specific datasets are constructed from this corpus as described below. In all cases, seven molecular properties are computed per molecule using RDKit (Landrum, 2006) for conditioning. Some of these descriptors are approximately additive over fragments, such as MW and hydrogen-bond counts, whereas others, such as logP, QED, TPSA, and RotBonds, depend more strongly on the assembled molecular structure and therefore provide complementary conditioning signals; Appendix E.4 summarizes each descriptor. Datasets are divided into train, validation, and test scaffold splits with validation and test held out, using an 8:1:1 ratio.

Molecule Generation. The molecule-generation datasets use fragment vocabularies of size $k \in \{100, 500, 1000, 2000\}$ and training corpus sizes $N \in \{1\text{k}, 10\text{k}, 100\text{k}, 1\text{M}\}$. *Scaffold Generation.* We extract Murcko scaffolds from the base molecules, yielding approximately 71M unique scaffolds. Following the protocol above, we use scaffold fragment vocabularies of size $k \in \{50, 100, 200, 500\}$ and training corpus sizes $N \in \{10\text{k}, 100\text{k}, 1\text{M}\}$. The smaller vocabulary range reflects the reduced fragment diversity in scaffold space. *Molecule Optimization.* We construct matched molecular pairs from the base corpus using Tanimoto similarity in the range $[0.70, 0.95]$, ensuring that each pair shares a common scaffold while differing by a small structural modification. Pairs are further filtered by aligning their BRICS fragment trees and requiring that the transformation corresponds to the insertion, deletion, or substitution of one or two fragments. Following the construction protocol above, we use fragment vocabularies of size $k \in \{500, 1000, 2000\}$ and training corpus sizes $N \in \{10\text{k}, 100\text{k}, 1\text{M}\}$. *Scaffold Decoration.* For scaffold decoration, each molecule is decomposed into a Murcko scaffold and the branches attached to that scaffold. We rank extracted branches by frequency and retain the top-k branches as the decoration vocabulary, with $k \in \{500, 1000, 2000\}$. We keep only molecules whose attached branches are all covered by this vocabulary, subsample $N \in \{10\text{k}, 100\text{k}, 1\text{M}\}$ molecules, and decompose the retained molecules into scaffold-plus-branch

training examples: the scaffold provides the fixed core for decoration, and the attached branches serve as the ground-truth retrieval targets.

3.2 Benchmarking Setup

We choose baselines with publicly available implementations that can be trained under a unified experimental protocol. Existing open-source methods primarily target molecule generation and molecule optimization; to the best of our knowledge, no publicly available baselines support scaffold generation or scaffold decoration. Accordingly, we benchmark Fraglingo against prior work on the two tasks where direct comparison is possible, while evaluating scaffold generation and scaffold decoration through comprehensive self-evaluation, including generation quality and controlled studies of vocabulary size and training corpus size. Every model is trained with three independent random seeds, and all reported results are the mean and standard deviation across seeds. All evaluated models, including Fraglingo, contain between 1.8M and 6.4M trainable parameters.

Property-Conditional Molecule Generation. We compare against MolGPT (Bagal et al., 2022), MolRWKV (Li et al., 2025), and PS-VAE (Kong et al., 2022). The selected baselines span complementary generation paradigms, including SMILES autoregression, graph-enhanced sequence generation, and fragment or subgraph latent-variable modeling. To ensure a controlled comparison, all trainable models are retrained on the same 10k-molecule training corpus used by Fraglingo. Fragment-based methods use a shared BRICS vocabulary of 500 fragments, while non-fragment methods are trained on the corresponding molecular dataset under their native representation. During evaluation, each model generates 20 molecules for each of 100 randomly sampled target-property vectors, conditioning jointly on seven molecular properties (Appendix E.4).

Molecule Optimization. We compare against HierVAE (Jin et al., 2020) and JT-VAE (Jin et al., 2018), retraining all methods under the same experimental protocol. All models are trained on the same 10k matched-pair dataset using a BRICS fragment vocabulary of 500 fragments, with an 8:1:1 scaffold split and three random seeds. Unlike unconditional property generation, optimization targets must remain chemically close to the reference molecule. Randomly sampling target-property vectors frequently produces unattainable objectives or requires large structural changes. Thus, for each held-out optimization pair, we use the original molecule together with the target properties of its paired optimized molecule as input. This guarantees that a chemically plausible solution exists while preserving the intended optimization setting and avoiding unrealistic property shifts.

3.3 Evaluation Metrics

Generation quality. We report four standard sanity-check metrics that assess whether a model produces valid, varied, and non-memorized molecules, independent of property control. *Validity* measures the fraction of generated outputs that parse as valid RDKit (Landrum, 2006) molecules. *Uniqueness* is the fraction of valid molecules with distinct SMILES strings. *Novelty* is the fraction of unique molecules absent from the training set. *Diversity* is defined as $1 - \overline{T}$, where $\overline{T}$ is the mean pairwise Tanimoto similarity over Morgan fingerprints (radius 2, 2048 bits), averaged within each condition batch and then across conditions.

Property control. We assess property control at the per-property, joint, and partial levels.

Single-property conditioning. Each of the seven molecular properties is evaluated independently by conditioning the model on a single target property. For each property, we measure the Spearman correlation between the requested target value and the value of the generated molecule computed using RDKit descriptors.

Normalized Joint Distance (NJD). The primary multi-property metric is the Normalized Joint Distance, defined per molecule as:

$$\text{NJD} = \sqrt{\sum_{p} \left(\frac{\hat{y}_p - y_p}{\sigma_p} \right)^2} \tag{2}$$

where $\hat{y}_p$ and y_p are the achieved and target values for property p, and σ_p is the training-set standard deviation of property p. Normalizing by σ_p prevents high-magnitude properties (e.g., MW in daltons) from dominating over unit-bounded ones (e.g., QED $\in [0, 1]$). Lower NJD is better.

Joint success rate. We report the fraction of generated molecules where all seven properties simultaneously fall within per-property absolute tolerances: logP ± 1.0, MW ± 50 Da, QED ± 0.15, TPSA ± 20,

HBD ± 1, HBA ± 2, RotBonds ± 2. We evaluate at two tolerance multipliers ($1\times, 2\times$); Joint@$1\times$ (unscaled) is the stricter number.

Partial satisfaction. Because the all-or-nothing joint metric can obscure near-misses, we also report the mean count of properties individually within their $1\times$ threshold (out of 7). This captures generations that are close but fail on a single property.

3.4 Results

Section 3.4.1 probes the learned representation and shows that Fraglingo captures an attachment-aware view of the molecular context. Section 3.4.2 shows that Fraglingo achieves the strongest joint property control among comparably trained baselines. Section 3.4.3 then demonstrates inference-time vocabulary expansion, the capability that continuous retrieval uniquely enables. Appendix A isolates the contribution of the continuous-retrieval objective. Appendix B, C and D show that the same primitive transfers to molecule optimization, scaffold generation, and scaffold decoration without modification.

3.4.1 The Wildcard Anchor Learns a Attachment-Aware Representation

The wildcard anchor is motivated by a simple hypothesis: for a fragment with multiple attachment sites, the readout should depend on the active site rather than only on the fragment identity. We test this directly using 400 asymmetric fragments with exactly two wildcards, which account for 40.5% of the 1000-fragment vocabulary. For each fragment, we compute two readouts, one using each wildcard as the query. If the wildcard-anchored readout ignores the choice of attachment site, the two embeddings should be identical. If it captures attachment-specific context, the embeddings should differ. Figure 3(a) visualizes the readouts with t-SNE, linking the two readouts of each fragment and coloring them alike. Many paired readouts are well separated, indicating that changing the active wildcard substantially alters the embedding. At the same time, the links are frequently near-parallel, suggesting that switching the active wildcard induces a consistent transformation in the learned embedding space rather than an arbitrary perturbation. Figure 3(b) makes this concrete: the fragments whose two readouts are most separated are those whose attachment sites sit in clearly different local environments, while the fragments whose readouts nearly coincide are close to symmetric. The readout is therefore not a noisy copy of a global fragment embedding; it carries a site-specific view of the same chemistry, separating attachment contexts precisely when the geometry warrants it and collapsing them when it does not.

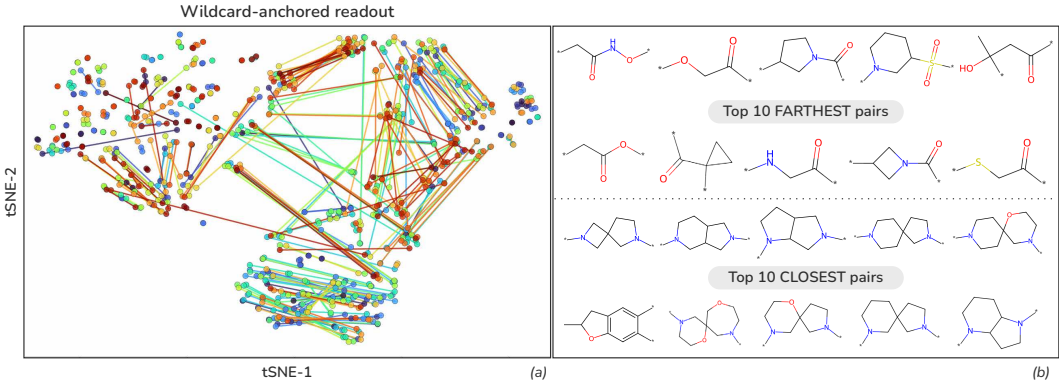


Figure 3: *(a)* tSNE visualization of wildcard-anchored readouts. Each fragment contributes two readouts, one per wildcard; paired readouts are linked and shown with the same color. *(b)* fragments with the largest and smallest distances between their two wildcard-anchored readouts. Farthest pairs have distinct attachment environments, whereas closest pairs are nearly symmetric.

3.4.2 Property-Conditional Generation Performance

We first evaluate property-conditional molecule generation against retrained baselines, then study how Fraglingo scales with training corpus and retrieval-vocabulary size.

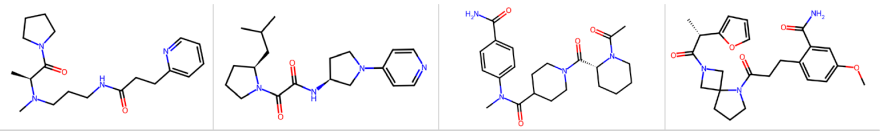
The difficulty of seven-way conditioning lies in satisfying all target properties simultaneously. Table 2 shows that Fraglingo with BFE fragmentation (Nguyen & Ji, 2026) gives the strongest joint control, with the lowest NJD (1.49), highest Joint@$1\times$ (69.65%), highest Joint@$2\times$ (97.75%), and

Table 2: Property-conditional molecule generation benchmarking results. Each model is followed by a row reporting the standard deviation over 3 runs where available. Best results are shown in **bold**; second-best results are underlined.

Model	Generation Quality				Joint Control				Per-Property Spearman							
	Validity (↑)	Uniqueness (↑)	Novelty (↑)	Diversity (↑)	NJD (↓)	Joint@1× (↑)	Joint@2× (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
MolGPT	25.7	**100.0**	**100.0**	0.3099	1.7339	53.97	95.02	6.32	0.8072	**0.9442**	0.6659	**0.8537**	0.7340	0.8242	0.6421	0.7816
(Std)	(1.9)	(0)	(0)	(0.026)	(0.113)	(1.362)	(3.945)	(0.190)	(0.038)	(0.031)	(0.074)	(0.037)	(0.057)	(0.039)	(0.056)	(0.012)
MolRWKV	90.9	**100.0**	**100.0**	0.8261	1.6090	43.77	90.46	6.17	0.8606	0.4830	**0.7070**	0.6449	**0.8839**	**0.8572**	**0.8232**	0.7514
(Std)	(2.2)	(0)	(0)	(0.027)	(0.103)	(4.851)	(4.196)	(0.161)	(0.032)	(0.044)	(0.075)	(0.040)	(0.041)	(0.030)	(0.023)	(0.027)
PS-VAE	**100.0**	99.8	**100.0**	**0.9029**	2.6985	1.33	22.82	3.55	0.4732	0.4774	0.0930	0.4100	0.1093	0.5485	0.1612	0.3246
(Std)	(0)	(0.1)	(0)	(0.029)	(0.118)	(0.658)	(1.861)	(0.174)	(0.087)	(0.111)	(0.183)	(0.058)	(0.061)	(0.051)	(0.036)	(0.063)
Fraglingo (BRICS)	**100.0**	**100.0**	**100.0**	0.8444	1.9808	40.88	85.90	5.80	0.8373	0.6423	0.5632	0.7605	0.8161	0.7800	0.7760	0.7393
(Std)	(0)	(0)	(0)	(0.027)	(0.054)	(2.111)	(3.602)	(0.191)	(0.035)	(0.033)	(0.042)	(0.066)	(0.046)	(0.081)	(0.043)	(0.038)
Fraglingo ablation†	**100.0**	99.8	**100.0**	0.8478	2.1256	34.66	81.90	5.63	0.7918	0.5128	0.4247	0.7254	0.8185	0.7377	0.6769	0.6697
(Std)	(0)	(0.8)	(0)	(0.028)	(0.060)	(2.34)	(3.81)	(0.204)	(0.037)	(0.036)	(0.045)	(0.064)	(0.047)	(0.079)	(0.045)	(0.041)
Fraglingo (BFE)	**100.0**	99.9	**100.0**	0.8218	**1.4928**	**69.65**	**97.75**	**6.56**	0.8710	0.7746	0.6373	0.8296	0.8284	0.8190	0.7993	**0.7942**

† Wildcard-anchored readout ablated.

highest average number of satisfied properties (6.56/7). Among the plain-BRICS settings, Fraglingo maintains 100% validity, 100% uniqueness, and 100% novelty while the wildcard-anchored readout ablation degrades joint control (NJD 2.1256 and Joint@1× 34.66%), supporting the role of the site-specific readout. MolGPT and MolRWKV are competitive on several aggregate metrics, whereas PS-VAE performs substantially worse under this stringent seven-property conditioning. Figure 4 gives representative conditional generation samples. The examples illustrate the model can satisfy multiple property targets while producing chemically varied structures rather than repeatedly returning the same fragment pattern.



	Target	Value	Error	Target	Value	Error	Target	Value	Error	Target	Value	Error
LogP	2	1.46	-0.537	2	1.81	-0.186	2	1.39	-0.612	1.92	2.33	+0.407
MW	350	346	-3.52	350	344	-5.54	400	415	+14.5	432	440	+7.15
QED	0.7	0.69	-0.012	0.7	0.85	+0.146	0.7	0.804	+0.104	0.73	0.713	-0.017
TPSA	70	65.5	-4.46	70	65.5	-4.46	100	104	+4.02	93.8	106	+12.3
HBD	1	1	0	1	1	0	1	1	0	0	1	+1
HBA	4	4	0	4	4	0	4	4	0	5	5	0
RB	_	9	_	4	4	0	4	4	0	7	7	0

Figure 4: Examples from property-conditional **molecule generation**. Green marks properties close to the requested target, and yellow marks properties within the acceptable tolerance range.

Table 3 separates corpus scale from retrieval-library size, and the two act differently. At 1k molecules, the model remains data-limited: the best setting reaches NJD 3.93 and Joint@1× 12.24, while larger vocabularies mainly increase diversity. Increasing the corpus to 10k sharply improves property control, with the 100-fragment model reaching NJD 1.85, Joint@1× 45.83, and Partial 5.95/7. At 100k molecules, the 1000-fragment model gives the strongest joint control, with NJD 1.52, Joint@1× 58.98, Joint@2× 96.45, and Partial 6.29/7, while the 500-fragment model achieves the highest average per-property correlation (0.8052). Scaling to 1M molecules does not uniformly improve over 100k; its best joint-control setting uses 500 fragments and reaches NJD 1.57 and Joint@1× 57.18. Overall, larger vocabularies consistently increase diversity, but property control is strongest when corpus scale and retrieval-library size are balanced.

3.4.3 Continuous Retrieval Enables Vocabulary Expansion After Training

The continuous-retrieval formulation enables a capability that classification-based generators cannot naturally support: because decoding is performed by nearest-neighbor search in a learned embedding space, the fragment library can be expanded after training without modifying or retraining the model. We evaluate this capability directly by training with one fragment vocabulary and retrieving from progressively larger libraries at inference time. Table 4 demonstrates this capability. Expanding the inference library causes the model to select many fragments that were never present in the training vocabulary, while preserving saturated validity, uniqueness, and novelty. This expansion slightly increases molecular diversity, but it also weakens property control: NJD rises and the average number of satisfied property targets decreases as the retrieval library grows. The same pattern appears for both the 500-fragment and 1000-fragment training settings. These results show that inference-time

Table 3: Property-conditional molecule generation under different vocabulary sizes and training dataset sizes. Best results are shown in **bold**; second-best results are underlined.

Train	Vocab	Generation Quality				Joint Control				Per-Property Spearman							
		Validity (↑)	Uniqueness (↑)	Novelty (↑)	Diversity (↑)	NJD (↓)	Joint@1× (↑)	Joint@2× (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
1K	100	97.8	98.7	**100.0**	0.8344	**3.9288**	**12.24**	**42.58**	**3.99**	0.5940	**0.2953**	0.2920	0.3903	**0.5228**	0.3101	0.4275	0.4045
	500	97.8	99.0	**100.0**	0.8631	4.2554	11.84	39.05	3.82	**0.6623**	0.1473	**0.3138**	**0.4428**	0.4705	0.3381	**0.4717**	**0.4066**
	1000	99.6	99.3	**100.0**	0.8748	4.1446	10.10	36.15	3.78	0.4501	0.1098	0.2547	0.4414	0.4598	**0.3536**	0.1643	0.3190
	2000	**99.8**	**99.6**	**100.0**	**0.8775**	4.3054	11.25	35.18	3.78	0.5069	0.1394	0.2646	0.3582	0.4166	0.2783	0.4005	0.3377
10K	100	**100.0**	**100.0**	**100.0**	0.8241	**1.8532**	**45.83**	**91.00**	**5.95**	0.7928	**0.7242**	**0.6436**	0.8212	0.7287	**0.8492**	**0.8188**	**0.7683**
	500	**100.0**	**100.0**	**100.0**	0.8444	1.9808	40.88	85.90	5.80	0.8373	0.6423	0.5632	0.7605	**0.8161**	0.7800	0.7760	0.7393
	1000	**100.0**	99.9	**100.0**	0.8484	1.8920	44.13	87.75	**5.95**	0.7655	0.5963	0.5495	**0.8245**	0.8059	0.7987	0.7738	0.7305
	2000	**100.0**	**100.0**	**100.0**	**0.8605**	2.0198	38.28	86.25	5.73	**0.8618**	0.6016	0.5292	0.7510	0.7924	0.7472	0.7023	0.7122
100K	100	**100.0**	**100.0**	**100.0**	0.8278	1.7492	51.83	91.15	6.06	0.7689	0.7362	0.5950	0.7323	0.6964	0.8063	0.7944	0.7327
	500	**100.0**	**100.0**	**100.0**	0.8499	1.5440	56.68	94.65	6.22	**0.8901**	0.7800	0.6361	**0.8456**	0.7890	**0.8577**	0.8382	**0.8052**
	1000	**100.0**	**100.0**	**100.0**	0.8526	**1.5189**	**58.98**	**96.45**	**6.29**	0.8524	0.7501	**0.6551**	0.8391	**0.8080**	0.8038	**0.8467**	0.7935
	2000	**100.0**	**100.0**	**100.0**	**0.8616**	1.6277	56.08	93.80	6.17	0.8819	**0.8026**	0.6500	0.8442	0.7981	0.8191	0.8380	0.8048
1M	100	**100.0**	**100.0**	99.8	0.8317	1.8422	52.10	88.79	5.98	0.7862	0.7703	0.6342	0.6829	0.6200	0.7767	0.7346	0.7149
	500	**100.0**	**100.0**	**100.0**	0.8525	**1.5668**	**57.18**	**94.25**	**6.20**	0.8658	0.8231	0.6249	0.7836	0.7710	0.8259	0.8172	0.7873
	1000	**100.0**	**100.0**	**100.0**	0.8567	1.6078	56.88	93.65	6.19	**0.8836**	0.7691	**0.6690**	**0.8012**	**0.7898**	**0.8298**	0.8280	**0.7957**
	2000	**100.0**	99.9	**100.0**	**0.8651**	1.6137	57.16	93.75	6.18	0.8739	**0.8546**	0.6451	0.7953	0.7718	0.7702	**0.8450**	0.7936

vocabulary expansion provides a practical, training-free mechanism for exploring a larger chemical space. The strongest property control is achieved when the retrieval library matches the training distribution, allowing practitioners to trade a modest loss in accuracy for greater chemical coverage without retraining.

Table 4: **Inference-time vocabulary expansion.** Models are trained with a fixed BFE fragment vocabulary but retrieve against a larger inference library at test time. OOV Frag.% is the fraction of selected fragments absent from the training vocabulary.

Inference Vocabulary	Generation Quality				Joint Control		Per-Property Spearman							Avg.	OOV Frag.
	Validity (↑)	Uniqueness (↑)	Novelty (↑)	Diversity (↑)	NJD (↓)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	(↑)	
Training (500-frags)	**100.0**	99.9	**100.0**	0.8218	1.4928	6.56	**0.8710**	0.7746	**0.6373**	0.8296	0.8284	0.8190	**0.7993**	0.7942	**0.00%**
Inference (1000-frags)	**100.0**	**100.0**	**100.0**	0.8313	1.6028	6.40	0.8600	0.7496	0.6323	**0.8476**	**0.8534**	**0.8280**	0.7903	**0.7945**	29.11%
Inference (2000-frags)	**100.0**	**100.0**	**100.0**	**0.8382**	1.8628	6.16	0.8620	0.7286	0.5953	0.8226	0.8334	0.8150	0.7773	0.7763	48.76%
Training (1000-frags)	**100.0**	**100.0**	**100.0**	0.8313	**1.2318**	**6.83**	0.7740	**0.8896**	0.6043	0.8456	0.7354	0.7890	0.7523	0.7702	**0.00%**
Inference (2000-frags)	**100.0**	**100.0**	**100.0**	0.8362	1.5018	6.54	0.7560	0.8676	0.6043	0.8376	0.7374	0.7855	0.7653	0.7648	28.32%

4 Related Work

Early molecular generators represent molecules as SMILES strings or molecular graphs. SMILES-based methods use recurrent networks, variational autoencoders, or Transformers (Segler et al., 2018; Gómez-Bombarelli et al., 2018; Ross et al., 2022), while graph-based methods generate atoms and bonds through autoregressive, flow, or diffusion processes (Shi et al., 2020; Hoogeboom et al., 2022). Despite architectural differences, these approaches largely operate at the atom or token level, leaving chemically meaningful substructures to be learned implicitly. Fragment-based methods instead generate larger chemical units that more closely reflect medicinal-chemistry operations. Methods including HierVAE (Jin et al., 2020), JT-VAE (Jin et al., 2018), PS-VAE (Kong et al., 2022), MoLeR (Maziarz et al., 2022), and FragGPT (Yue et al., 2024) generate discrete fragments from fixed vocabularies, while t-SMILES (Wu et al., 2024) and SynCoGen (Rekesh et al., 2025) explore hierarchical and synthesis-oriented fragment representations. These methods nevertheless rely on discrete fragment or building-block vocabularies, limiting expansion to unseen fragments. Fraglingo instead uses attachment-aware continuous latent retrieval, allowing the fragment library to expand at inference time while conditioning retrieval on the active attachment site.

5 Conclusion, Limitations, and Future Work

Fraglingo introduces a retrieval-based formulation for fragment-level molecular design. Instead of classifying over fixed fragment identifiers, the model predicts a continuous fragment embedding and decodes it by nearest-neighbor retrieval. Combined with a wildcard-anchored readout that conditions retrieval on the active attachment site, this formulation jointly determines fragment identity and attachment while enabling inference-time expansion of the fragment vocabulary without retraining. Across molecule generation, scaffold generation, scaffold decoration, and molecule optimization, the same latent retrieval architecture is reused by changing only the retrieval query and conditioning signal. Fraglingo remains limited by the chosen fragmentation scheme and vocabulary. Future work includes richer fragmentation strategies, stronger conditioning mechanisms, and scaling retrieval to substantially larger fragment libraries.

REFERENCES

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Molgpt: molecular generation using a transformer-decoder model. *Journal of chemical information and modeling*, 62(9):2064–2076, 2022.

Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

Jorg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503–1507, 2008.

Carl Edwards, Chi Han, Gawon Lee, Thao Nguyen, Sara Szymkuc, Chetan Kumar Prasad, Bowen Jin, Jiawei Han, Ying Diao, Ge Liu, Hao Peng, Bartosz A. Grzybowski, Martin D. Burke, and Heng Ji. mclm: A modular chemical language model that generates functional and makeable molecules. In *Proc. The Fourteenth International Conference on Learning Representations (ICLR2026)*, 2026.

Nathan C Frey, Ryan Soklaski, Simon Axelrod, Siddharth Samsi, Rafael Gomez-Bombarelli, Connor W Coley, and Vijay Gadepally. Neural scaling of deep chemical models. *Nature Machine Intelligence*, 5(11):1297–1305, 2023.

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.

Philip J Hajduk and Jonathan Greer. A decade of fragment-based drug design: strategic advances and lessons learned. *Nature reviews Drug discovery*, 6(3):211–219, 2007.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1): 015022, 2022.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pp. 4839–4848. PMLR, 2020.

Philine Kirsch, Alwin M Hartman, Anna KH Hirsch, and Martin Empting. Concepts and core principles of fragment-based drug design. *Molecules*, 24(23):4309, 2019.

Xiangzhe Kong, Wenbing Huang, Zhixing Tan, and Yang Liu. Molecule generation by principal subgraph mining and assembling. *Advances in Neural Information Processing Systems*, 35: 2550–2563, 2022.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Greg Landrum. RDKit: Open-source cheminformatics. `https://www.rdkit.org`, 2006.

Thomas R Lane, Patricia A Vignaux, Joshua S Harris, Scott H Snyder, Fabio Urbina, and Sean Ekins. Machine learning and large language models for modeling complex toxicity pathways and predicting steroidogenesis. *Environmental science & technology*, 59(27):13844–13856, 2025.

Xihan Li, Kuanping Gong, Yongquan Jiang, Yan Yang, and Tianrui Li. Molrwkv: Conditional molecular generation model using local enhancement and graph enhancement. *Journal of Computational Chemistry*, 46(10):e70100, 2025.

Krzysztof Maziarz, Henry 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. arXiv:2103.03864.

Thao Nguyen and Heng Ji. Mollingo: Molecule-native representations for llm-powered scientific agents. *arXiv preprint arXiv:2605.27853*, 2026.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Andrei Rekesh, Miruna Cretu, Dmytro Shevchuk, Vignesh Ram Somnath, Pietro Liò, Robert A Batey, Mike Tyers, Michał Koziarski, and Cheng-Hao Liu. Syncogen: Synthesizable 3d molecule generation via joint reaction and coordinate modeling. *arXiv preprint arXiv:2507.11818*, 2025.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12):1256–1264, 2022.

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1): 120–131, 2018.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Morgan Thomas, Mazen Ahmad, Gary Tresadern, and Gianni De Fabritiis. Promptsmiles: prompting for scaffold decoration and fragment linking in chemical language models. *Journal of Cheminformatics*, 16(1):77, 2024.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Juan-Ni Wu, Tong Wang, Yue Chen, Li-Juan Tang, Hai-Long Wu, and Ru-Qin Yu. t-smiles: a fragment-based molecular representation framework for de novo ligand design. *Nature Communications*, 15(1):4993, 2024.

Jie Yue, Bingxin Peng, Yu Chen, Jieyu Jin, Xinda Zhao, Chao Shen, Xiangyang Ji, Chang-Yu Hsieh, Jianfei Song, Tingjun Hou, et al. Unlocking comprehensive molecular design across all scenarios with large language model and unordered chemical language. *Chemical Science*, 15(34): 13727–13740, 2024.

A Ablations Attribute the Gains to Continuous Retrieval

Section 3.4.2 shows that Fraglingo achieves strong property control. Here, we isolate the contribution of each design component by holding the setting fixed (10k corpus, 500 fragments, 6.4M parameters) and changing one component at a time:

- **Contrastive vs. regression objective.** Replace InfoNCE with a plain regression loss ($\|\hat{z} - e_2\|^2$) to the true fragment embedding, keeping nearest-neighbor decoding. This asks whether the retrieval geometry needs in-batch negatives or only a target direction.
- **Classification loss.** Remove $\mathcal{L}_{\text{cls}}$ to test the embedding-collapse hypothesis of Section 2.1: without a discriminative pressure on fragment embeddings, InfoNCE alone may map distinct fragments to overlapping regions.
- **RVQ regularizer.** Remove $\mathcal{L}_{\text{commit}}$ and the codebook to test whether RVQ further organizes the latent space beyond the classification objective, encouraging compact and retrievable fragment embeddings.

Table 5: Component ablation on the shared 10k-molecule, 500-fragment setting. Each row removes or replaces exactly one component relative to the full model (top row); rows are ordered by severity of the resulting drop in joint control.

Configuration	Generation Quality				Joint Control				Per-Property Spearman							
	Validity (↑)	Uniqueness (↑)	Novelty (↑)	Diversity (↑)	NJD (↓)	Joint@1× (↑)	Joint@2× (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
Full model (Fraglingo)	**100.0**	99.9	**100.0**	0.8218	**1.4928**	**69.65**	**97.75**	**6.56**	**0.8710**	0.7746	**0.6373**	**0.8296**	**0.8284**	**0.8190**	**0.7993**	**0.7942**
InfoNCE (→ MSE regression)	99.8	**100.0**	**100.0**	0.8238	4.2248	26.85	24.94	3.73	0.1010	0.3146	0.0033	0.3636	0.1674	0.1470	0.1453	0.1772
Classification loss	**100.0**	**100.0**	**100.0**	0.8228	1.7878	56.52	88.14	5.95	0.7570	0.6966	0.4823	0.7756	0.7434	0.7520	0.7193	0.7032
RVQ regularizer	**100.0**	**100.0**	**100.0**	**0.8268**	1.5268	67.75	95.76	6.29	0.7780	**0.8536**	0.6053	0.8046	0.7344	0.7980	0.7243	0.7562

The ablation cleanly attributes the model's control to the continuous-retrieval objective. Replacing InfoNCE with direct MSE regression is catastrophic: NJD rises from 1.4928 to 4.2248, Joint@1× drops from 69.65% to 26.85%, and average per-property correlation falls from 0.7942 to 0.1772, essentially destroying property control while generation quality stays intact. Regressing to a target point is not enough; the contrastive objective, with its in-batch negatives, is what organizes the embedding space into a geometry where nearest-neighbor retrieval selects the right fragment. This is the core mechanism behind the first contribution, and the ablation shows it is not optional but load-bearing.

The two regularizers play supporting roles consistent with their design intent. Removing the classification loss degrades control (NJD 1.7878, Joint@1× 56.52%, Avg. 0.7032), confirming the embedding-collapse hypothesis: without a discriminative pressure, distinct fragments blur together and retrieval loses precision. Removing RVQ has the mildest effect (NJD 1.5268, Joint@1× 67.75%), so RVQ is a light stabilizer rather than a primary signal; we keep it because it costs little and slightly improves control, but the method does not depend on it.

B One Primitive Transfers to Molecule Optimization

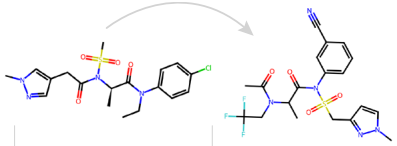
The next three sections evaluate Fraglingo beyond molecule generation, showing how the retrieval mechanism extends to molecule optimization, scaffold generation, and scaffold decoration by changing only the query and conditioning signals. We begin with molecule optimization.

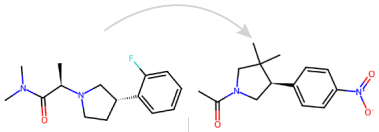
We benchmark our Fraglingo's molecule optimization model against HierVAE and JT-VAE under the shared matched-pair optimization protocol. All optimization models are trained on the same 10k matched-pair dataset; fragment-based models use a vocabulary of 500 fragments. As described in Section 2.2, reference-molecule conditioning turns Fraglingo's attachment-conditioned retrieval mechanism into a molecule optimizer; here it gives the strongest target matching among the compared fragment-based methods. Table 6 shows Fraglingo attains the lowest final NJD (1.6244), preserves the starting molecule best among the compared methods (Tanimoto 0.7952 and 95.72% above 0.6 similarity), and gives the best average distributional similarity. Fraglingo makes similarity-constrained edits when needed to hit multi-property targets, which is the intended behavior for property-directed optimization.

Table 6: Molecule optimization against fragment-based baselines on the shared 10k matched-pair dataset. Tanimoto and Sim≥0.6 measure preservation of the starting molecule.

Model	Generation Quality		Optimization Performance				Property Distribution Similarity							
	Validity (↑)	Novelty (↑)	NJD (↓)	Tanimoto (↑)	Sim≥0.6 (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
HierVAE	**100.0%**	**100.0%**	1.7717	0.6746	66.14%	**6.17**	0.821	0.723	**0.727**	0.719	0.817	0.809	0.719	0.762
(Std)	(0)	(0)	(0.011)	(0.014)	(2.6)	(0.006)	(0.010)	(0.012)	(0.009)	(0.010)	(0.014)	(0.004)	(0.004)	(0.011)
JT-VAE	**100.0%**	99.9%	2.6481	0.2940	4.12%	5.19	0.509	0.329	0.373	0.277	0.537	0.421	0.515	0.423
(Std)	(0)	(0)	(0.093)	(0.027)	(0.74)	(0.60)	(0.078)	(0.069)	(0.090)	(0.063)	(0.093)	(0.071)	(0.102)	(0.098)
Fraglingo	**100.0%**	**100.0%**	**1.6244**	**0.7952**	**95.72%**	**6.17**	**0.839**	**0.744**	0.713	**0.772**	**0.869**	**0.840**	**0.788**	**0.795**
(Std)	(0)	(0)	(0.018)	(0.008)	(1.0)	(0.010)	(0.004)	(0.007)	(0.004)	(0.014)	(0.011)	(0.003)	(0.006)	(0.009)

Figure 5 shows representative optimization examples produced by Fraglingo. The examples highlight the intended edit behavior: Fraglingo changes local fragments to move properties toward the target while preserving much of the starting molecule.





	Target	Value	Error	Value	Error	Target	Value	Error	Value	Error
LogP	2	1.85	-0.154	1.95	-0.046	1.69	2.09	+0.402	2.57	+0.677
MW	500	427	-73.1	344	-28.5	273	264	-8.66	262	-10.7
QED	0.7	0.67	-0.027	0.85	-0.091	0.82	0.834	+0.014	0.607	-0.213
TPSA	120	92.6	-27.4	65.5	-3.63	70	23.6	-46.4	63.5	-6.55
HBD	0	0	0	1	0	0	0	0	0	0
HBA	7	6	-1	4	0	3	2	-1	3	0
RB	7	7	0	4	0	2	3	+1	2	0

Figure 5: Qualitative **molecule optimization** examples. Green marks properties close to target, yellow marks properties within tolerance, and red marks properties outside the target range.

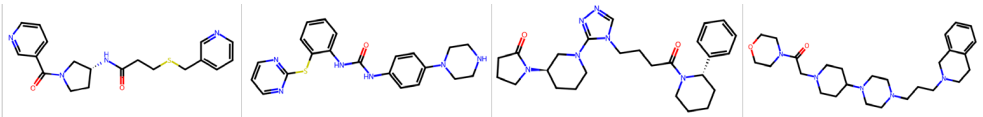
Table 7: Ablation on training corpus size and vocabulary size for molecule optimization. Tanimoto and Sim≥0.6 measure preservation of the starting molecule.

Train	Vocab	Generation	Optimization Performance				Property Distribution Similarity							
		Validity (↑)	NJD (↓)	Tanimoto (↑)	Sim≥0.6 (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
10k		**100.0%**	1.6244	0.7952	95.72%	6.17	0.8390	0.7440	0.7130	0.7720	0.8690	0.8400	0.7880	0.7950
100k	500	**100.0%**	1.4031	0.7903	**95.82%**	6.34	0.8709	0.7712	0.7681	0.8327	0.8753	0.8593	0.8103	0.8269
1M		**100.0%**	**1.2037**	0.7853	94.62%	**6.39**	0.8617	0.7830	**0.7981**	0.8351	0.8941	**0.8927**	**0.8791**	0.8491
10k		**100.0%**	1.6477	0.7896	94.72%	6.12	0.8414	0.6757	0.7280	0.7772	0.8656	0.8328	0.7924	0.7876
100k	1000	**100.0%**	1.4429	0.7747	94.42%	6.24	0.8558	0.7664	0.7629	0.8146	0.8549	0.8227	0.8330	0.8158
1M		**100.0%**	1.2820	0.7653	93.52%	6.30	0.8519	**0.8127**	0.7867	**0.8537**	0.8906	0.8881	0.8668	**0.8501**
10k		**100.0%**	1.6232	**0.7980**	95.02%	6.13	0.8273	0.7136	0.7478	0.7639	0.8731	0.8258	0.8426	0.7992
100k	2000	**100.0%**	1.4840	0.7774	93.12%	6.20	0.8241	0.7196	0.7489	0.8201	0.8788	0.8508	0.8360	0.8112
1M		**100.0%**	1.3082	0.7669	92.82%	6.31	**0.8747**	0.7695	0.7831	0.8488	**0.8965**	0.8646	0.8716	0.8441

Table 7 shows that increasing the matched-pair training corpus consistently improves target matching, whereas the effect of vocabulary size is more nuanced. The 1M-pair, 500-fragment model achieves the lowest NJD (1.2037), while the 1M-pair, 1000-fragment model gives the highest average property-distribution similarity (0.8501). Larger vocabularies can improve preservation for smaller training sets—the 10k-pair, 2000-fragment model attains the highest Tanimoto similarity (0.7980)—but do not consistently improve target matching as the corpus grows. All configurations maintain 100% validity, indicating that training-set coverage is the primary driver of optimization performance once the vocabulary provides sufficient edit diversity.

C ... TO SCAFFOLD GENERATION

We evaluate scaffold generation through controlled self-evaluation because, to the best of our knowledge, no available scaffold-generation baseline can be retrained under the same protocol. The evaluation therefore focuses on whether Fraglingo, when retargeted to scaffold fragments, produces valid ring systems and diverse core topologies (Figure 6), and on how property control varies with vocabulary and corpus size (Table 8). Scaffold generation can be viewed as a constrained version of full-molecule generation, with retrieval limited to scaffold fragments.



	Target	Value	Error	Target	Value	Error	Target	Value	Error	Target	Value	Error
LogP	2	2.13	+0.131	3.82	3.68	-0.139	3.7	3.4	-0.297	0	1.38	+1.38
MW	350	370	+20.5	429	407	-22.6	486	465	-21.8	486	470	-16.7
QED	0.7	0.756	+0.056	0.58	0.561	-0.019	0.55	0.628	+0.078	0.55	0.601	+0.051
TPSA	70	75.2	+5.19	79.8	82.2	+2.39	76.4	74.6	-1.81	50	42.5	-7.5
HBD	1	1	0	3	3	0	1	0	-1	1	0	-1
HBA	4	5	+1	6	6	0	6	6	0	6	6	0
RB	_	7	_	4	5	+1	7	7	0	7	7	0

Figure 6: Qualitative **scaffold generation** examples. Fraglingo assembles scaffold-level BRICS fragments into valid core structures before side-chain decoration, producing diverse ring systems and attachment-ready topologies rather than complete decorated molecules.

Table 8: Scaffold generation under different vocabulary sizes and training dataset sizes.

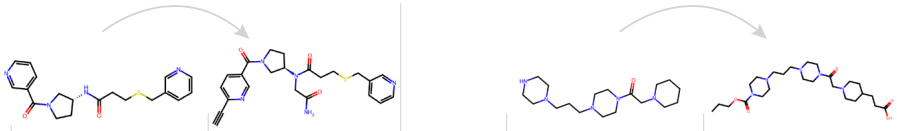
Train	Vocab	Generation Quality				Joint Control				Per-Property Spearman							
		Validity (↑)	Uniqueness (↑)	Novelty (↑)	Diversity (↑)	NJD (↓)	Joint@1× (↑)	Joint@2× (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
10k	50	**100.0**	96.8	88.2	0.8059	2.5943	41.67	84.64	5.68	0.6169	**0.7711**	0.4029	0.5954	0.6196	0.5987	0.6701	0.6107
	100	**100.0**	97.0	89.4	0.8262	**2.5278**	**44.22**	**84.99**	**5.77**	**0.7397**	0.7302	0.3994	0.6019	0.5859	0.6052	**0.7786**	**0.6344**
	200	**100.0**	99.0	95.2	0.8363	3.1045	32.05	74.45	5.28	0.7253	0.6606	0.2815	0.5352	**0.6709**	0.4859	0.5754	0.5621
	500	99.9	**99.2**	**97.0**	0.8397	3.9841	25.28	63.91	4.74	0.6526	0.4954	0.1639	0.4121	0.6040	0.2993	0.6391	0.4666
100k	50	99.9	98.8	82.8	0.8082	2.7067	37.04	78.83	5.48	0.5840	0.5958	0.3421	0.5226	0.4922	0.5371	0.5929	0.5238
	100	99.9	99.4	90.0	0.8219	2.4809	42.94	81.58	5.65	0.6564	0.6445	**0.4560**	**0.6031**	0.6055	**0.6137**	0.6474	0.6038
	200	99.3	99.3	93.6	**0.8404**	3.0413	33.79	71.50	5.22	0.6512	0.6276	0.3659	0.5720	0.5105	0.4341	0.7090	0.5529
	500	99.9	**99.6**	**97.0**	**0.8464**	3.4155	31.83	64.71	5.06	0.5277	0.4496	0.3100	0.5591	0.6119	0.4730	0.6463	0.5111
1M	50	**100.0**	99.3	80.7	0.8097	2.4646	39.67	83.19	5.69	0.6037	0.6050	0.3359	0.5659	0.5024	0.5036	0.5694	0.5266
	100	**100.0**	99.4	91.2	0.8141	**2.1501**	**46.45**	**87.05**	**5.94**	**0.7783**	**0.6790**	**0.4996**	**0.6379**	0.5802	**0.6558**	0.7109	**0.6488**
	200	**100.0**	**99.8**	93.6	0.8333	2.4190	41.62	81.79	5.65	0.6885	0.6271	0.4551	0.5838	0.5940	0.5915	0.7098	0.6071
	500	99.9	99.7	**95.6**	**0.8441**	2.9393	33.88	70.67	5.29	0.6526	0.5193	0.3484	0.5514	**0.6140**	0.5585	**0.7671**	0.5730

Table 8 shows a stronger vocabulary-size tradeoff for scaffold-only generation than for full-molecule generation, and this pattern remains visible at the 1M training-corpus scale. Small vocabularies, especially 100 fragments, give the best joint-control metrics because they concentrate probability mass on common scaffold motifs and make the retrieval problem easier. Larger vocabularies increase uniqueness, novelty, and diversity, but the additional rare fragments worsen NJD and joint satisfaction performance. This behavior is expected for scaffolds: the model must control global core topology with fewer substituent degrees of freedom, so an overly broad scaffold vocabulary can make property-conditioned retrieval less reliable.

D ... AND TO SCAFFOLD DECORATION

Scaffold decoration is again evaluated by self-comparison across settings rather than against external systems, for the same protocol-compatibility reason. Prompt-based decoration methods such as PromptSMILES (Thomas et al., 2024) use different conditioning interfaces, while scaffold-extension methods such as MoLeR are primarily optimized for structural plausibility rather than explicit multi-property control. The question here is whether site-conditioned retrieval chooses chemically plausible branches for a fixed core while keeping scaffold atom indices stable across sites.

Figure 7 illustrates the behavior of the decoration model on fixed scaffolds with explicit property targets. Because each attachment site is encoded from the viewpoint of the corresponding scaffold



	Target	Value	Error	Value	Error	Target	Value	Error	Value	Error
LogP	2	2.13	+0.13	1.31	-0.69	2	-0.09	-2.09	1.26	-0.74
MW	425	370	-55	452	+26.6	_	338	_	496	_
QED	0.6	0.756	+0.156	0.455	-0.145	_	0.734	_	0.458	_
TPSA	110	75.2	-34.8	109	-0.51	110	42.1	-67.9	96.9	-13.1
HBD	1	1	0	1	0	_	1	_	1	_
HBA	4	5	+1	6	+2	_	5	_	7	_
RB	8	7	-1	9	+1	_	6	_	11	_

Figure 7: Qualitative **scaffold decoration** examples. For each fixed scaffold, Fraglingo predicts branch fragments at the growable sites under the specified property targets. The tables report molecular-property values and signed errors before and after decoration, illustrating how site-conditioned retrieval can move decorated molecules toward the requested property profile while preserving the scaffold core.

atom, the model selects different branch fragments for different local environments while leaving the core unchanged. Retrieved decorations reduce errors on targeted properties such as TPSA and molecular weight, though improving one property can trade off against weakly constrained descriptors. This matches the intended use of decoration as a controlled local edit: the scaffold fixes the conserved core, and Fraglingo searches the branch-embedding space for site-compatible substituents aligned with the requested profile.

Table 9 quantifies this over the held-out decoration set. We report the same generation-quality and joint-control metrics as for molecule generation, computed on decorated molecules while the scaffold is held fixed.

Table 9: **Scaffold decoration** under different vocabulary and corpus sizes. Metrics are computed on decorated molecules with the scaffold held fixed.

Train	Vocab	Joint Control			Per-Property Spearman							
		NJD (↓)	Tanimoto (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
10k	500	1.08	0.3165	6.62	0.9126	0.7919	0.8381	0.8632	0.8996	0.8538	0.8032	0.8518
	1000	0.95	0.3338	6.75	0.9115	0.8711	0.8852	0.9194	0.8839	0.9225	0.8891	0.8975
	2000	1.00	0.3206	6.70	0.9030	0.8290	0.8287	0.8965	0.8770	0.9069	0.8870	0.8755
100k	500	0.80	0.3349	6.75	0.9412	0.8488	**0.9086**	**0.9469**	0.8826	0.9324	0.8958	0.9080
	1000	0.81	0.3503	6.73	0.9370	0.8687	0.8599	0.9137	0.8873	0.9273	0.9049	0.8998
	2000	**0.71**	**0.3565**	**6.82**	0.9387	**0.8966**	0.9062	0.9377	**0.9173**	**0.9458**	**0.9327**	**0.9250**
1M	500	0.98	0.3373	6.59	0.9226	0.8359	0.8515	0.8974	0.8653	0.9062	0.8639	0.8775
	1000	0.83	0.3445	6.76	0.9291	0.8553	0.8658	0.9332	0.9089	0.9266	0.9224	0.9059
	2000	0.83	0.3313	6.75	0.9376	0.8622	0.8924	0.9351	0.8953	0.9366	0.9212	0.9115

Table 9 shows that decoration remains a highly controlled local-edit task. Increasing the corpus from 10k to 100k consistently improves the aggregate metrics: NJD decreases, Tanimoto similarity increases, partial satisfaction stays near or above 6.7 out of 7, and the average per-property correlation rises for every vocabulary size. The best overall setting is 100k training molecules with a 2000-fragment vocabulary, which gives the lowest NJD (0.71), highest Tanimoto similarity (0.3565), highest partial satisfaction (6.82/7), and highest average correlation (0.9250). A larger 1M corpus remains competitive but does not improve the best aggregate result, indicating that more data is not the main bottleneck once the decoration model has enough matched scaffold–branch coverage. Vocabulary size has a milder effect than corpus scale: larger vocabularies help most at 100k, but the trend is not monotonic at 10k or 1M. Across settings, high correlations for TPSA, HBA, HBD, and rotatable bonds indicate that site-conditioned retrieval can steer decorated molecules toward the requested property profile while preserving the fixed scaffold core.

E Detailed Fraglingo Model Design

The appendix expands the concise method description in Section 2. We first specify the autoregressive training examples, then detail the encoder, readout, conditioning, predictor, RVQ regularizer, losses, metrics, and model scale.

E.1 Generation Framing

We decompose each molecule into fragments using BRICS bond-breaking rules, yielding a fragment tree with associated attachment-point wildcards. For each molecule, we identify the main chain as the longest path in this fragment tree and create two next-fragment prediction trajectories, one starting from each end of the chain. Within each trajectory, autoregressive training examples are ordered by a breadth-first traversal that starts from the selected terminal cap fragment, defined as a fragment with exactly one wildcard.

At step t, the model receives the partial assembly G_t with one designated active wildcard atom and must predict the embedding of the fragment that attaches at that site. A terminal expansion is represented by attaching a cap fragment with a single wildcard, which consumes the active wildcard without introducing another growable site. Molecule growth stops when the assembly has no remaining wildcard atoms available for expansion.

E.2 Graph Encoder: FragmentGPS

All graph-structured inputs, including both partial assemblies and individual fragments, are encoded by a single shared encoder ϕ, FragmentGPS. FragmentGPS is a 6-layer GPS (Graph + Transformer hybrid) network with hidden dimension $d = 256$.

Each GPSLayer contains three sub-modules applied sequentially with residual connections and LayerNorm:

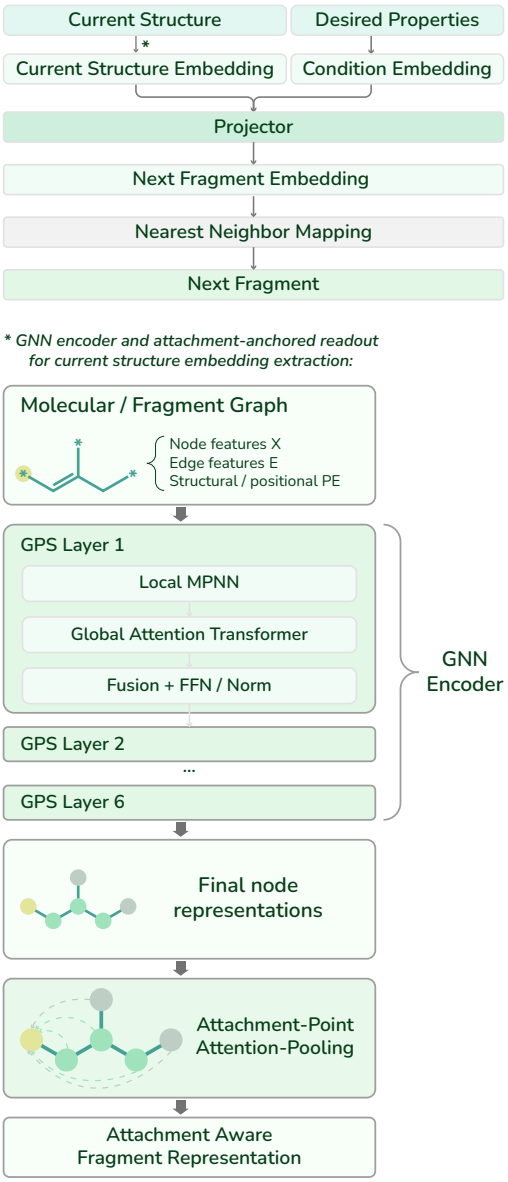


Figure 8: Detailed architecture of Fraglingo's next-fragment generation module.

1. **Local message passing** via GINE (GIN with edge features): each atom aggregates messages from its bonded neighbors, where bond-type features are incorporated into the aggregation.
2. **Global self-attention**: multi-head self-attention ($h = 8$ heads) over all atoms in the graph, giving each atom access to global context regardless of graph distance.
3. **Feed-forward block**: a position-wise MLP ($d \to 4d \to d$, GELU activation).

Dropout of 0.1 is applied throughout.

Attachment-anchored readout. Standard graph-level pooling (mean or sum over atoms) is replaced by an attachment-anchored readout: the graph embedding is computed as the attention-weighted sum of context-atom representations, where the query vector comes from the active attachment atom. For autoregressive generation this query atom is the open wildcard; for scaffold decoration it is the active scaffold atom. Formally, let h_q denote the active query atom's final-layer hidden state and

$H = [h_1, \ldots, h_n]$ the matrix of context-atom hidden states. The graph embedding is:

$$e = \mathrm{softmax}\left(\frac{h_q H^\top}{\sqrt{d}}\right) H \tag{3}$$

This anchors the assembly representation to the current attachment point rather than averaging over the whole molecule. For terminal cap-fragment prediction, the active wildcard atom's embedding is used directly as the query.

E.3 Property Conditioning

Seven molecular property descriptors are computed per molecule: logP, molecular weight (MW), QED, topological polar surface area (TPSA), hydrogen bond donors (HBD), hydrogen bond acceptors (HBA), and the count of rotatable bonds. Each is z-scored using per-dataset statistics computed over the training set.

The property vector $\mathbf{p} \in \mathbb{R}^7$ is linearly projected to $\mathbb{R}^d$ and added to the partial-assembly embedding before the predictor MLP:

$$e_1^{\text{cond}} = e_1 + W_p \mathbf{p} + \mathbf{b}_p \tag{4}$$

This is an additive conditioning injection. During training, $\mathbf{p}$ is zeroed with probability $p_{\text{drop}} = 0.2$ independently per training example, jointly training a conditional and an unconditional model in a single pass. This enables classifier-free guidance at sampling time, where the final predicted embedding is interpolated as:

$$\hat{z}_{\text{guided}} = (1 + w)\, \hat{z}_{\text{cond}} - w\, \hat{z}_{\text{uncond}} \tag{5}$$

for guidance scale $w \geq 0$.

E.4 Conditioning Property Descriptors

We condition on seven RDKit descriptors that cover complementary aspects of molecular design:

- **logP** estimates octanol–water partitioning and serves as a proxy for hydrophobicity and membrane permeability.
- **MW** is molecular weight, controlling overall molecule size and often changing approximately additively with fragment additions.
- **QED** summarizes drug-likeness by combining several physicochemical preferences into a single score.
- **TPSA** measures topological polar surface area, reflecting polar atom exposure and transport-related behavior.
- **HBD** and **HBA** count hydrogen-bond donors and acceptors, which affect polarity and binding interactions.
- **RotBonds** counts rotatable bonds and acts as a compact measure of molecular flexibility.

MW, HBD, and HBA are largely determined by the fragments present, whereas logP, QED, TPSA, and RotBonds also depend strongly on how fragments are assembled. Together, these descriptors test whether the model can control both fragment-compositional and structure-dependent properties.

E.5 Predictor: $\text{MLP}_{\text{Mapper}}$

The conditioned context embedding e_1^{cond} is passed through $\text{MLP}_{\text{Mapper}}$, a two-layer MLP:

$$\hat{z} = W_2 \cdot \mathrm{GELU}(W_1\, e_1^{\text{cond}} + b_1) + b_2 \tag{6}$$

with hidden dimension $4d = 1024$. The output $\hat{z} \in \mathbb{R}^d$ is the predicted embedding of the next fragment, in the same space as fragment embeddings produced by ϕ.

E.6 Discrete Bottleneck: RVQCodebook

A 4-stage residual vector quantizer (RVQ) operates on the predicted embedding $\hat{z}$ during training. Each stage maintains a codebook of 64 vectors updated via exponential moving averages (EMA), with dead-code reset when a code goes unused for too many steps. Given $\hat{z}$, stage 1 finds the nearest codebook vector c_1 and passes the residual $\hat{z} - c_1$ to stage 2; after stage m, the residual is $\hat{z} - \sum_{\ell=1}^{m} c_\ell$. The resulting 4-tuple of code indices (k_1, k_2, k_3, k_4) spans $64^4 \approx 16$M possible combinations.

The RVQ is a training-time regularizer only and plays no role in decoding. At inference, the normalized predicted embedding $\hat{z}$ is compared directly against a precomputed table of normalized fragment embeddings $\{\phi(f)\}_{f \in \mathcal{V}}$ via inner product, which is equivalent to cosine similarity, and the fragment with the highest similarity is selected. The table can be recomputed for a larger inference vocabulary than the one used to form training targets, allowing vocabulary expansion without changing the predictor or adding a new output head.

E.7 Training Objective

The model is trained end-to-end with three losses, weighted by hyperparameters:

$$\mathcal{L} = \lambda_{\text{NCE}} \mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{commit}} \mathcal{L}_{\text{commit}} \tag{7}$$

We select these weights by grid search and use $\lambda_{\text{NCE}} : \lambda_{\text{cls}} : \lambda_{\text{commit}} = 1 : 0.5 : 0.25$ in the reported experiments.

InfoNCE loss. For a batch of B training steps with ℓ_2-normalized predicted embeddings $\{\hat{z}_i\}$ and ℓ_2-normalized true next-fragment embeddings $\{e_{2,i}\}$, the InfoNCE loss is:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\hat{z}_i \cdot e_{2,i}/\tau)}{\sum_{j=1}^{B} \exp(\hat{z}_i \cdot e_{2,j}/\tau)} \tag{8}$$

using in-batch negatives. This is the primary signal driving the predictor to point toward the correct fragment in embedding space.

Fragment classification loss. A linear classification head is applied to the true fragment embedding $e_2 = \phi(f_{t+1})$ and trained with cross-entropy over the vocabulary. This auxiliary loss discourages embedding collapse: without it, InfoNCE alone may not keep fragment embeddings sufficiently discriminative, and the encoder can map multiple fragments to the same region of embedding space.

VQ commit loss. The commit loss pulls the continuous prediction $\hat{z}$ toward its nearest codebook vector:

$$\mathcal{L}_{\text{commit}} = \left\| \hat{z} - \text{sg}(\hat{z}_{\text{quantized}}) \right\|^2 \tag{9}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. Codebook vectors are updated via EMA rather than backpropagation.

E.8 Evaluation Metrics

Three metrics are tracked during training:

- **acc(e_2)**: fragment classifier accuracy on the true next-fragment embedding e_2, serving as a sanity check that the encoder produces discriminative representations across the vocabulary.
- **acc($\hat{z}$@1)**: top-1 accuracy of the predicted embedding $\hat{z}$ under the fragment classifier, measuring whether the prediction lands on the correct next fragment. This is the primary generation accuracy metric compared across ablation arms.
- **acc($\hat{z}$@5)**: top-5 accuracy of $\hat{z}$ under the same classifier.

E.9 Model Scale

The full model has approximately 6.4M trainable parameters. The dominant components are the 6-layer FragmentGPS encoder (shared across partial assemblies and fragments), the $\text{MLP}_{\text{Mapper}}$

Model	Dim.	Layers	Heads	Params
Small	128	4	4	**1.2M**
Baseline	256	6	8	**6.4M**
Large	512	8	8	**32.9M**

Table 10: Model configurations and parameter counts.

predictor head, and the RVQ codebook parameters. Table 10 summarizes the small, baseline, and large model configurations used to assess the effect of model scale.

Table 11 reports property-conditional generation across model sizes and fragmentation methods. Scaling from 1.2M to 6.4M parameters yields modest improvements in joint control (NJD $2.011 \rightarrow 1.981$, Joint@$1\times$ $38.9 \rightarrow 40.88$, Joint@$2\times$ $84.9 \rightarrow 85.90$), while generation quality remains saturated. Further scaling to 32.9M does not improve joint control (NJD 2.017, Joint@$1\times$ 39.05, Joint@$2\times$ 83.70, Avg. 0.733), suggesting that this 10k-molecule, 500-fragment benchmark is not primarily capacity-limited. We also report a variant (6.4M*) that keeps the architecture fixed but applies BPE on BRICS fragments, called BFE (BRICS-based Fragment Enumeration) in MolLingo (Nguyen & Ji, 2026), coarsening the fragment vocabulary into fewer, larger composite units. This variant substantially improves joint control (NJD 1.493, Joint@$1\times$ 69.65, Joint@$2\times$ 97.75, Partial 6.56) and achieves the highest average per-property Spearman correlation (0.794), showing that Fraglingo can benefit substantially from coarser fragmentation schemes. Thus, the plain-BRICS results reported in this work should not be interpreted as an upper bound on the retrieval framework. We keep plain BRICS fragments as the default because they are well-established and widely used; MolLingo-style BFE (BPE-on-BRICS) is an orthogonal tokenization change that composes with the same retrieval primitive, and we report it as evidence that the method can benefit from coarser vocabularies rather than as a separate model.

Table 11: Effect of model size on property-conditional molecule generation (10k corpus, 500 fragments). Scaling parameters improves joint control at saturated generation quality. The 6.4M* row applies BPE on BRICS fragments, called BFE (BRICS-based Fragment Enumeration) in MolLingo, coarsening the vocabulary.

Model size	Generation Quality				Joint Control				Per-Property Spearman							
	Validity (↑)	Uniqueness (↑)	Novelty (↓)	Diversity (↑)	NJD (↓)	Joint@1× (↑)	Joint@2× (↑)	Partial (/7) (↑)	logP (↑)	MW (↑)	QED (↑)	TPSA (↑)	HBD (↑)	HBA (↑)	RotBonds (↑)	Avg. (↑)
1.2M	**100%**	**100%**	**100%**	0.843	2.011	38.9%	84.9%	5.80	0.823	0.611	0.568	0.756	0.821	0.776	0.756	0.730
6.4M	**100%**	**100%**	**100%**	**0.8444**	1.9808	40.88%	85.90%	5.80	0.8373	0.6423	0.5632	0.7605	0.8161	0.7800	0.7760	0.7393
32.9M	**100%**	**100%**	**100%**	0.8443	2.017	39.05%	83.70%	5.73	0.793	0.621	0.603	0.754	0.825	0.782	0.750	0.733
6.4M*	**100%**	99.9%	**100%**	0.8218	**1.4928**	**69.65%**	**97.75%**	**6.56**	**0.8710**	**0.7746**	**0.6373**	**0.8296**	**0.8284**	**0.8190**	**0.7993**	**0.7942**

* Same architecture as 6.4M, but tokenization applies BPE on BRICS fragments, called BFE (BRICS-based Fragment Enumeration) in MolLingo, rather than using BRICS fragments alone.

F Baseline Experimental Setup

Shared property protocol. All baselines use the same 7-property target vector as Fraglingo: logP, molecular weight, QED, TPSA, HBD, HBA, and rotatable bond count, with all labels computed by RDKit. Unless a baseline required a different data interface, we trained it from scratch on the same approximately 10k-molecule corpus used for the Fraglingo conditional generator, using a 90/10 train/validation split. Conditional-generation methods were evaluated on the same 100 held-out property targets, with 20 samples drawn for each target. Optimization methods were evaluated on 200 held-out source molecules paired with target 7-property vectors and scored using the same normalized joint-distance protocol as Fraglingo.

We selected baselines according to two criteria: (1) the official implementation provides a reproducible training and evaluation pipeline, and (2) the method addresses at least one of the four molecular design tasks considered in this work, namely property-conditional generation, molecule optimization, scaffold generation, or scaffold decoration. Methods whose official implementations do not support end-to-end reproduction were excluded. For example, although FragGPT (Yue et al., 2024) is closely aligned with the tasks studied here, its public repository does not provide a complete training pipeline, preventing a fair comparison. We likewise excluded methods designed solely for unconditional

molecular generation (such as ChemGPT (Frey et al., 2023), MolBART (Lane et al., 2025), and Chemformer (Irwin et al., 2022)), as they cannot be directly adapted to our task formulations.

Fraglingo. All Fraglingo variants share the same backbone architecture. The model uses a shared FragmentGPS graph encoder with a residual vector-quantized (RVQ) codebook consisting of 4 stages with 64 codes each, a commitment weight of $\beta = 0.25$, and a classification loss weight of 0.5. Unless otherwise specified, architecture hyperparameters are fixed at $d_{\text{model}} = 256$, 6 GNN layers, 8 attention heads, and a dropout rate of 0.1. Models are trained using AdamW (learning rate 3×10^{-4}, weight decay 10^{-2}) with cosine annealing, gradient clipping at 1.0, and a batch size of 32. Training runs for up to 50 epochs with early stopping (patience (S)= 3, activated after the first epoch) based on validation top-1 acc(z@1).

F.1 Conditional generation baselines.

MolGPT. We used MolGPT (Bagal et al., 2022), a GPT-style autoregressive SMILES transformer, from the authors' official implementation[1] without architectural changes. The model is a decoder-only causal transformer with 8 layers, 8 attention heads, and embedding dimension 256. Following the original conditioning scheme, the 7-property vector is linearly projected to a single embedding token and prepended to the SMILES sequence before the causal self-attention blocks. We trained MolGPT with AdamW, cosine learning-rate decay, and batch size 256. At inference, SMILES were sampled autoregressively with ancestral sampling at temperature 1.0 and no top-k filtering until an end-of-sequence token was emitted.

MolRWKV. We used MolRWKV (Li et al., 2025), a SMILES language model based on the linear-attention RWKV-6 architecture, from the authors' official implementation[2] without architectural changes. The only local change was a path-resolution fix for locating custom CUDA kernel sources at build time. The model uses a 6-layer RWKV decoder with embedding dimension 256 over a 25-symbol character-level SMILES vocabulary. Conditioning is handled by a 2-layer RWKV property encoder: each scalar property is projected to the model dimension, producing 7 property-token embeddings that are encoded and injected into every decoder block. We trained MolRWKV with Adam, learning rate 5×10^{-4}, batch size 128, and early stopping with patience 5 for up to 50 epochs. At inference, SMILES were sampled autoregressively until an end-of-sequence token was emitted.

PS-VAE. We used PS-VAE (Kong et al., 2022), a principal-subgraph variational autoencoder that assembles molecules from automatically mined fragment pieces, from the authors' official implementation[3]. Local changes were limited to PyTorch Lightning API-compatibility updates and property-scoring utilities. The released code supports QED, synthetic accessibility, and penalized logP; we added RDKit scoring and normalization ranges for molecular weight, TPSA, HBD, HBA, and rotatable bond count, and replaced penalized logP with plain RDKit logP. The encoder, piece-based VAE decoder, and property-predictor MLP were otherwise unchanged. The model used latent dimension 56 and a 500-piece vocabulary mined from the training corpus. We jointly trained the property predictor to regress all 7 properties with Adam, learning rate 10^{-3}, batch size 32, and early stopping with patience 3 for up to 8 epochs. Because PS-VAE is not end-to-end conditional, inference used post-hoc latent optimization. For each target, we sampled 40 prior latents, optimized each for up to 50 Adam steps at learning rate 0.05 against the min-max-normalized property target with patience 10, and decoded optimized latents until 20 valid candidates were collected.

F.2 Optimization baselines.

HierVAE. We used the HierVAE (Jin et al., 2020) hierarchical graph-to-graph translation model (HierVGNN), which encodes and decodes molecular graphs over automatically extracted structural motifs, using the authors' official implementation[4]. Local changes were limited to robustness fixes: a guard against division-by-zero when preprocessing small datasets and numpy/tensor-to-float coercions for logging. We trained the translator variant with hidden and embedding size 270, latent size 4, and graph/tree message-passing depth 20. Training used the same matched-molecule-pair corpus

[1] https://github.com/devalab/molgpt
[2] https://github.com/bigwestHan/MolRWKV
[3] https://github.com/THUNLP-MT/PS-VAE
[4] https://github.com/wengong-jin/hgraph2graph

as Fraglingo optimization: 10k source-target pairs from the full matched-pairs dataset. Molecules incompatible with the codebase's motif extraction were filtered out. We trained with Adam, learning rate 10^{-3}, batch size 32, 20 epochs, and KL weight 0.3 annealed at rate 0.9. At inference, each held-out source used 20 stochastic translations over multiple motif attachment roots.