Exploring sampling optimizations and conditional generation in discrete flow matching for graphs

Beñat Solaegui [*1] **Ángel Oroz** [*1] **Asier Yániz** [*1]

Abstract

DeFoG is a recent flow-based framework for graph generation that introduces a key methodological innovation: the decoupling of training hyperparameters from sampling hyperparameters. This separation allows the model to achieve competitive results while enabling substantial improvements by modifying only the sampling configuration.

In this project, we aim to understand and extend DeFoG by first reproducing the main experiments reported in the original paper and then exploring several enhancements. Specifically, we introduce and compare alternative distortion functions, implement both structural and property-based conditional generation (including subgraph-fixing and attribute constraints), and conduct an extensive study of sampling hyperparameters showing that optimal settings are strongly step-dependent. These additions expand the original framework and provide deeper insight into how controllability and sampling design influence the quality of the generated graphs.

1. Introduction

Before deepening into our reproduction of the DeFoG experiments and the extensions we propose, we first want to introduce a few concepts; all of them are explained in detail in the original DeFoG paper (1).

Flow matching has recently emerged as a powerful framework for generative modeling, providing a unified perspective for transport-based generation methods (2)

The main contribution of DeFoG is that its sampling hyperparameters are fully decoupled from training. Once the model is trained, its sampling dynamics can be reshaped without modifying or retraining the network. In this project we focus on three such hyperparameters:

- **Distortion functions**: The distortion function controls how sampling steps are allocated across time, changing the pace at which noise is removed.
- **Target guidance** (ω): scales the deterministic update toward the model's predicted clean graph, controlling how strongly the sampler follows the learned direction.
- **Stochasticity** (η): controls the level of random perturbation added at each step, affecting the variability and exploration of generated samples.

These three parameters play different roles and together shape the quality, diversity, and stability of the generated graphs. Prior work on discrete normalizing flows has highlighted the importance of carefully designing the transformation space when modeling combinatorial structures (3). To assess the impact of these choices, we rely on standard metrics in graph generation. Validity measures the fraction of samples that conform to the structural rules of the dataset, while uniqueness quantifies how many of the valid graphs are distinct from each other, and novelty evaluates which of these unique samples do not appear in the training set. In our evaluation we primarily report two metrics:

- **V.U.N**: the proportion of generated graphs that are valid, unique and novel.
- **MMD Ratio**: the average ratio between the distances (e.g., MMDs) computed on graph statistics of the generated vs. test sets and those of the train vs. test sets.

Knowing the basic concepts of the DeFoG framework, we now outline the main components of our work. We begin with a reproducibility study in which we reconstruct the core experiments from the original paper. We then introduce new distortion functions, implement both structural and property-based conditional generation, and finally optimize sampling-specific hyperparameters to improve generation quality. All these experiments are carried out using different datasets, which are described in Appendix A.

[1]Student of the Swiss Joint Master in Computer Science, Bern, Switzerland. Correspondence to: Beñat Solaegui <benat.solaeguigarralda@students.unibe.ch>, Ángel Oroz <angel.orozazcarate@students.unibe.ch>, Asier Yániz <asier.yanizibanez@students.unibe.ch>.

2. Reproduction of Original Results

We have systematically reproduced all the key experiments reported in the original DeFoG paper, including the results summarized in Table 1, Figure 2a, and Figure 3. Across all experiments, we were able to successfully replicate the qualitative behaviour and trends of the original results, confirming the validity of the methodological approach. Early generative models such as GraphRNN paved the way for learning graph distributions by autoregressively modeling node and edge sequences (4).

2.1. Reproduction of Table 1

Table 1 evaluates the validity, uniqueness, and novelty of the generated graphs across different datasets and sampling steps, following the setup of the original DeFoG paper.

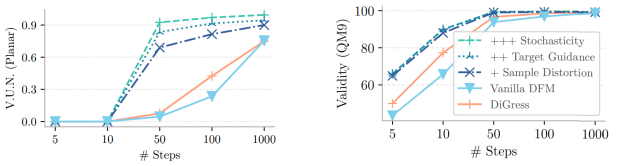
In our reproduction, we observed a consistent offset in some of the reported **MMD Ratio** values. This difference is attributable to discrepancies between the checkpoints we used for reproduction and those used in the original paper. To illustrate this effect, Table 1 contrasts three sets of results: the original results reported in the paper, our reproductions using the newest published checkpoints, and our reproductions using an older checkpoint version.

Although the absolute MMD Ratio values differ slightly across checkpoints, the results most similar to the original were obtained with the oldest checkpoints. However, since we initially ran everything with the newest ones, the rest of the experiments are based on the newest checkpoints, which is why the ratio may be slightly off in some plots, although the overall shape is preserved.

2.2. Reproduction of Figure 2a

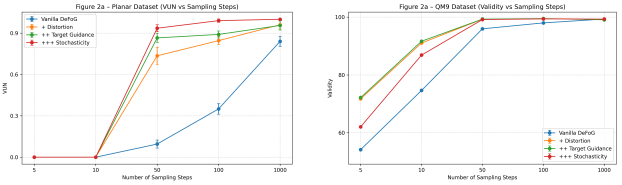
Figure 2a illustrates how generation quality evolves as the number of sampling steps increases. It also highlights the contribution of each hyperparameter, added sequentially across the curves in the plot. For planar graphs, we track the V.U.N. (Validity, Uniqueness, Novelty) metric, while for QM9 molecules, we monitor validity. The curves demonstrate that increasing the number of sampling steps generally improves generation quality, and that incorporating additional hyperparameters progressively enhances performance, emphasizing the impact of iterative refinement.

Overall, our reproduced results in Figures 2 closely match the trends observed in the original curves, Figures 1. This confirms that the generation process behaves consistently across both datasets, with increased steps and refined hyperparameters leading to higher validity and V.U.N. scores.



(a) Original Planar V.U.N. vs Steps (b) Original QM9 Validity vs Steps

Figure 1: Original curves from Figure 2a.

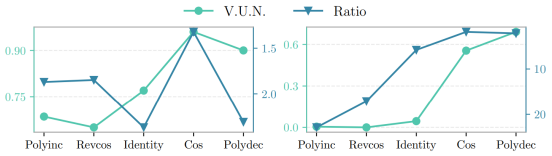


(a) Reproduced Planar V.U.N. vs Steps (b) Reproduced QM9 Validity vs Steps

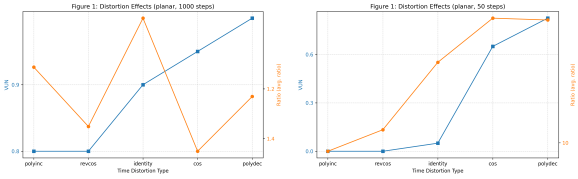
Figure 2: Reproduced curves for Figure 2a.

2.3. Reproduction of Figure 3

Figure 3 investigates how different sampling configurations affect generation quality, including variations in sample distortion, target guidance, and stochasticity, evaluated at 50 and 1000 steps.



(a) Original distortion experiment



(b) 1000 steps (Reproduced) (c) 50 steps (Reproduced)

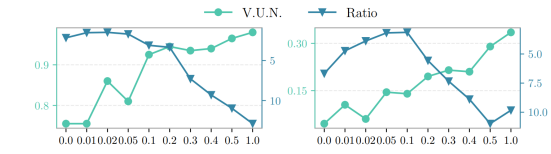
Figure 3: Reproduction of the distortion experiment.

As it is shown in Figure 3, Figure 4 and Figure 5, our results are similar to the original ones.

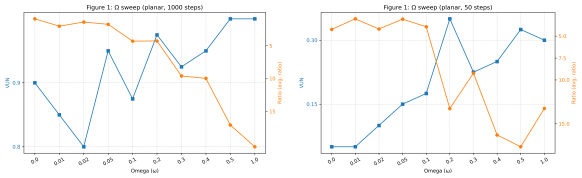
These results confirm that more extreme distortion worsens performance, target guidance modulates stability, and stochasticity introduces controlled diversity, in line with the original paper's observations.

Table 1: Comparison between the original paper results and our reproductions using the newest and older checkpoint versions for the original Table 1. Metrics are shown for each dataset at 50 and 1000 sampling steps.

Table	Steps	Planar		Tree		SBM	
		V.U.N. ↑	Ratio ↓	V.U.N. ↑	Ratio ↓	V.U.N. ↑	Ratio ↓
Original Results	50	95.0±3.2	3.2±1.1	73.5±9.0	2.5±1.0	86.5±5.3	2.2±0.3
	1000	99.5±1.0	1.6±0.4	96.5±2.6	1.6±0.4	90.0±5.1	4.9±1.3
Newest Checkpoint	50	93.5±2.6	6.0±1.8	74.5±4.0	2.1±0.7	83.5±7.2	7.0±2.5
	1000	100.0±0.0	2.9±0.9	95.0±3.9	2.0±0.3	85.5±2.9	6.7±0.7
Oldest Checkpoint	50	93.5±4.6	3.2±1.0	78.5±8.2	2.0±0.3	79.5±4.9	4.2±0.9
	1000	98.5±2.0	2.1±0.8	96.0±3.4	2.2±0.7	87.0±2.5	4.5±1.0

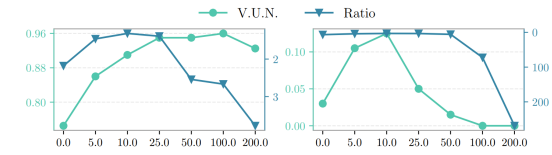


(a) Original target guidance experiment

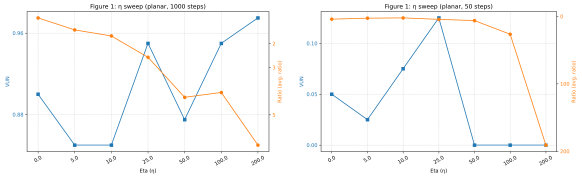


(b) 1000 steps (Reproduced) (c) 50 steps (Reproduced)

Figure 4: Reproduction of the target guidance experiment.



(a) Original stochasticity experiment



(b) 1000 steps (Reproduced) (c) 50 steps (Reproduced)

Figure 5: Reproduction of the stochasticity experiment.

3. Exploring Different Distortion Families

In this section, we explore three different families of temporal distortion functions. As mentioned, these functions reshape the sampling timeline by modifying how much change is applied at each denoising steps. For certain datasets, it is beneficial to apply smaller updates in the later stages of sampling. Large modifications near the end can break validity constraints at a point where the sampler can no longer recover. In addition, all distortion functions considered in this work are required to be monotonically increasing and to map the interval [0,1] onto [0,1], ensuring a valid and well-behaved reparameterization of the sampling timeline.

The original DeFoG paper already employs some distortion functions, but to continue studying their effect, we consider three broader families of distortions, each parametrized by a scalar a.

3.1. New Polydec Family

The first family, `new_polydec_a` (Equation 1), is a variant of the standard polynomial decay. This function differs mathematically from the original one but its grafical representation is quite similar. The parameter a allow us to interpolate between milder and more aggressive function shapes, as shown in Figure 6.

$$f(t) = 1 - (1 - t)^a \tag{1}$$

3.2. New Cosine-Like Family

The second family, `new_cos_a`, mimics the behavior of the cosine time distortion function. As we can appreciate in Figure 7 (Equation 2), this function allows for slight variations depending on the parameter a, being a mathematically different formula and having graphical similarities with cosine.

$$f(t) = \frac{t^a}{t^a + (1 - t)^a} \tag{2}$$

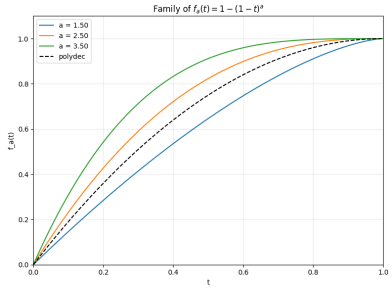


Figure 6: Values of the `new_polydec_a` function for different a parameters.

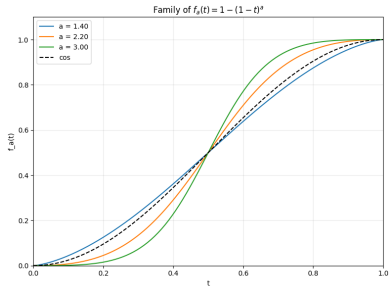


Figure 7: Values of the `new_cos_a` function for different a parameters.

3.3. Logarithmic-Like Family

The third family, `log_a` (Equation 3), is fundamentally different from the previous two as Figure 8 shows. Its purpose is to explore alternative behaviors in the temporal distortion.

$$f(t) = \frac{\log(1 + at)}{\log(1 + a)} \tag{3}$$

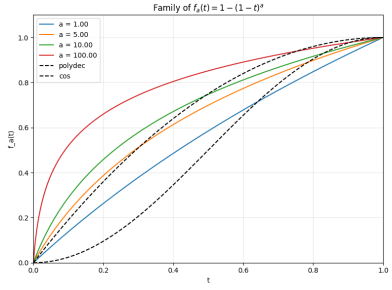


Figure 8: Values of the `log_a` function for different a parameters.

3.4. Comparison of V.U.N. values

Finally, we compare the resulting V.U.N. values for these three families against the previously used time distortions. As shown in Figure 9, the first two families (`new_polydec_a` and `new_cos_a`) exhibit overall similar behavior to the original time distortions, although their performance can vary depending on the choice of parameters. Notably, for the planar dataset with 50 sampling steps, `new_polydec_a` (with $a = 3.5$) can slightly outperform the original distortions. This improvement appears to come from using a flatter end of the curve, which reduces step size at the final stages and helps preserve validity constraints. In contrast, the `log_a` family follows a clearly different dynamic and consistently achieves lower V.U.N. values.

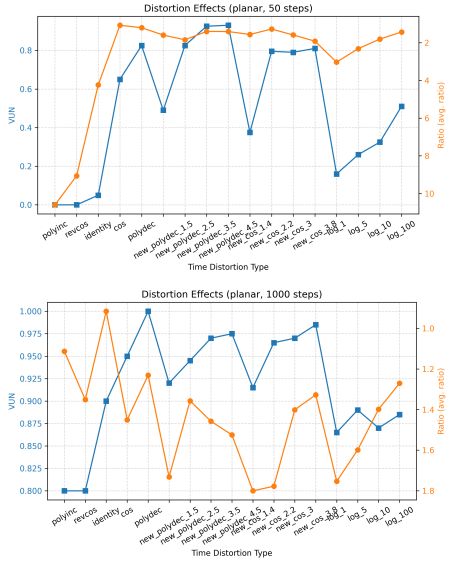


Figure 9: Comparison of average validity for different distortion families at 50 steps (top) and 1000 steps (bottom). The new families show different behaviors, while maintaining trends similar to the original schedules.

4. Conditional Generation

Conditional generation aims to guide the graph generation process based on additional constraints or prior knowledge. While the original DeFoG paper only briefly explores this concept on the TLS dataset, our work focuses primarily on extending and evaluating conditional generation deeper. For the implementation, we were inspired by other recent works in graph generation, such as the DiGress paper (5) or the Prodigy paper (6). And flow-based molecular generators such as GraphAF have shown the effectiveness of autoregressive flows for capturing complex chemical structures (7).

4.1. Subgraph-Conditioned Generation

In *subgraph-conditioned generation*, instead of starting from pure noise, a subset of nodes and edges is fixed from

the beginning, serving as a scaffold for the rest of the graph. During each generation step, the model produces the full graph as usual, but after the step we enforce the presence of the specified subgraph by overwriting the corresponding entries in the adjacency and node feature matrices. This ensures that the fixed subgraph remains intact throughout the generation process, while the remaining nodes and edges are free to adapt and connect with the fixed structure naturally. Importantly, this procedure is fully optional and can be applied to any dataset used in our study. A minimal example of this procedure is shown in Figure 10. Full-scale versions and additional examples are provided in Appendix B.

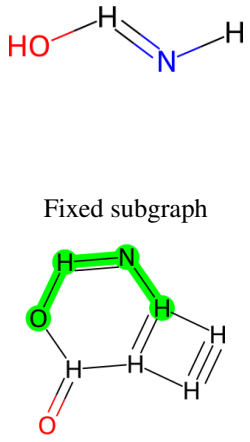


Figure 10: Illustration of subgraph-conditioned generation. The fixed subgraph (top) is extracted from a validation graph (middle) and enforced during the generative process (bottom).

Main Experiment We evaluate this approach on the *Planar* and *QM9* datasets. These choices are motivated by their simplicity and fast execution (planar), as well as the potential insight into molecular structures (QM9). For both datasets, we generate subgraphs or submolecules of varying sizes, derived from graphs in the validation set. For the *Planar* dataset, we fixed between 2 and 32 nodes, while for *QM9* we fixed 2 to 8 atoms. Figures 11 and 12 summarize the results in terms of validity compared to the number of fixed nodes.

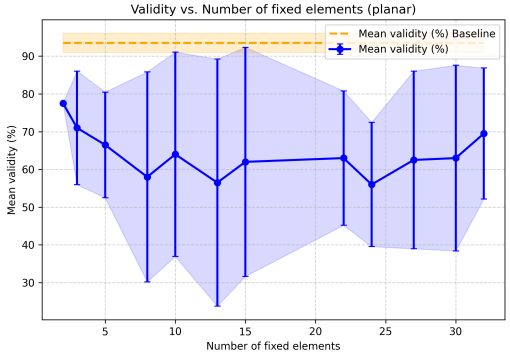


Figure 11: Validity versus the number of fixed nodes for Planar. Dashed line: unconstrained baseline. Fluctuations indicate that validity depends on subgraph structure rather than size.

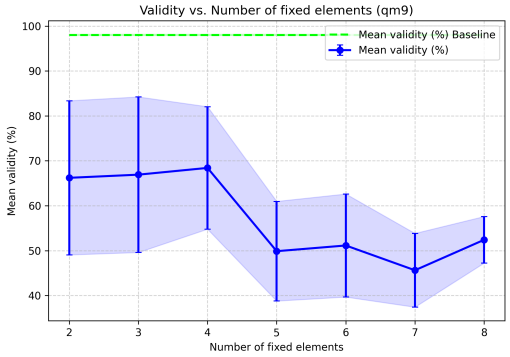


Figure 12: Validity versus the number of fixed atoms for QM9. Dashed line: unconstrained baseline. Validity decreases as more atoms are fixed, reflecting chemical constraints.

Analysis. For *Planar*, we observe that validity exhibits large variability both for small and large fixed-subgraph sizes. Some short subgraphs lead to high validity, while others with the same number of nodes collapse rapidly; the same occurs for larger subgraphs, where results range from excellent to poor. This heterogeneous behavior suggests that validity is not primarily determined by the size of the fixed subgraph, but rather by its intrinsic structure (e.g., compactness, closure, branching).

For *QM9*, the overall trend shows a clearer decline in validity as more atoms are fixed. This agrees with intuition: molecules impose strong chemical and functional constraints, making it harder for the model to complete larger

fixed fragments while preserving validity. Nonetheless, the decrease is not strictly monotonic and still shows variability across subgraphs of identical size. This again indicates that subgraph size alone does not fully explain validity.

Follow-up Experiment To further investigate whether validity is determined by the size of the fixed fragment or by its structural nature, we conduct a complementary experiment for both datasets. For *Planar*, we select a fixed-subgraph size with high variance in the previous results (15 nodes) and construct several subgraphs of identical size but different structure. For *QM9*, we follow the same protocol using fixed sub-molecules of size 2, chosen because this setting also exhibited heterogeneous outcomes despite the overall decreasing trend with increasing fixed atoms.

For each dataset, we evaluate five folds for: (i) the baseline (no conditioning), (ii) a "best-case" subgraph of the chosen size, and (iii) a "worst-case" subgraph of the same size.

Figures 13 and 14 show the resulting boxplots for Planar and QM9, respectively.

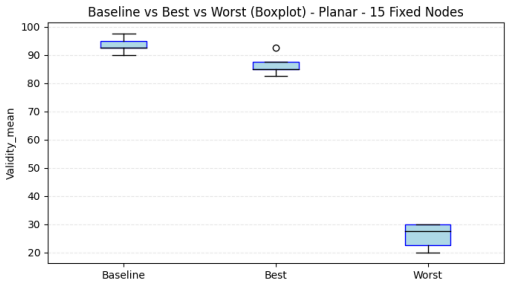


Figure 13: Planar dataset: V.U.N. distribution across 5 folds for the baseline and for the best and worst 15-node fixed subgraphs.

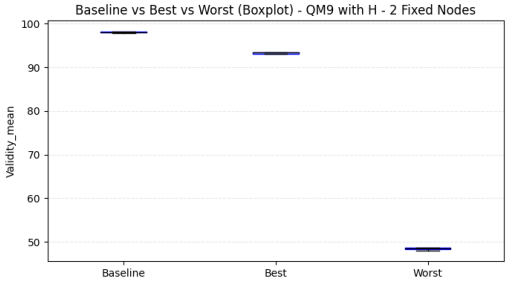


Figure 14: QM9 dataset: Validity across 5 folds for the baseline and for the best and worst 2-atom fixed sub-molecules.

Analysis. Across both datasets, the behavior is remarkably consistent. In the *best-case* scenarios, validity (or V.U.N. for Planar) remains extremely close to the baseline, and in some folds even surpasses it. This demonstrates that fixing an appropriate subgraph can guide generation without impairing quality, and may even provide a beneficial structural bias.

In contrast, the *worst-case* subgraphs produce drastically lower validity in both datasets, despite having the same number of fixed nodes or atoms as their best-case counterparts. The gap between best and worst cases is large and stable across folds, clearly indicating that the determining factor is not the fixed-subgraph size but the intrinsic structural nature of the fixed fragment.

Although QM9 exhibits a global downward trend when increasing the number of fixed atoms, the strong divergence between best and worst 2-atom sub-molecules confirms that structural compatibility still plays a major role. That is, while fixing more atoms generally restricts molecular flexibility and harms validity, the chemistry of the specific fragment remains a decisive factor.

These experiments consistently support the view that conditional generation is highly sensitive to the structural characteristics of the fixed subgraph or sub-molecule, rather than its size alone.

4.2. Property-Conditioned Generation

Beyond structure, conditional generation can be guided by target molecular properties, especially on datasets that provide this information, such as *QM9*. Here, molecules are generated to match a target property value, such as a specific energy. Implementing this required retraining the model from scratch, as the original DeFoG code does not support conditional generation out of the box.

Additionally, we needed to address an issue in the conditional sampling logic: in the original implementation, only the unconditional rate matrix was used during sampling. Correcting this ensured that the sampler properly incorporates the conditional guidance during generation.

Figure 15 illustrates the accuracy of this conditioning. The x-axis corresponds to the target property values, and the y-axis shows the properties of generated molecules. Ideally, points lie on the diagonal, indicating perfect adherence to the target.

To verify that conditioning does not compromise generation quality, we also plot validity for the property-conditioned molecules, Figure 16. Results indicate that enforcing property constraints does not degrade validity, maintaining standards comparable to the baseline.

Generally speaking, conditional generation in DeFoG can successfully guide graph or molecular generation either by fixing subgraphs/sub-molecules or by targeting specific molecular properties. Subgraph conditioning shows that the structural characteristics of the fixed fragment influence

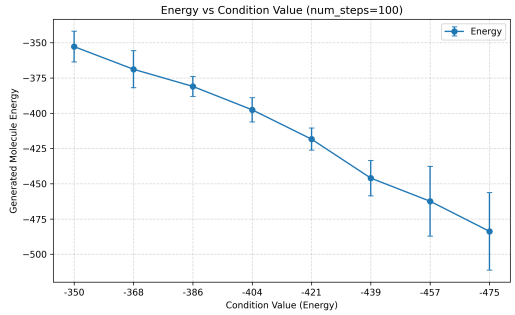


Figure 15: Target versus generated energy values for QM9 molecules under property-conditioned generation. Points closely align with the diagonal, demonstrating accurate control.

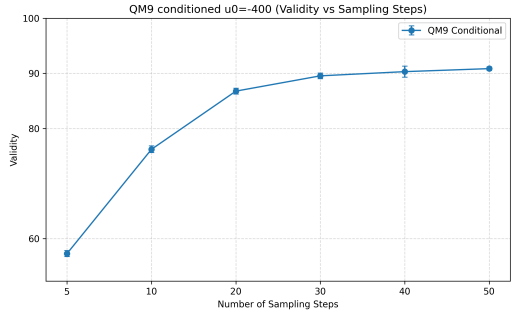


Figure 16: Validity of QM9 molecules under property-conditioned generation. Dashed line: unconstrained baseline. Property conditioning maintains high validity.

validity, while property conditioning achieves accurate control over molecular attributes without reducing generation quality.

5. Hyperparameter Optimization Across Sampling Steps

In principle, the optimal sampling hyperparameters for a given experiment also depend on the number of sampling steps. For instance, with a low number of steps, a small η may be advantageous, whereas a larger η typically improves performance when the number of steps is sufficiently high. However, the original paper uses the same sampling configuration across all step counts in their experiments. While they identify the best configuration per experiment, they do not investigate how the optimal hyperparameters vary with the number of sampling steps.

This observation motivates a natural question: *should sampling hyperparameters be optimized as a function of the number of steps?*

To address this, we perform a systematic hyperparameter–optimization study, aiming to identify the best parameters $(\eta, \omega, \text{distortion})$ conditioned on the sampling step count S. Our procedure consists of three main stages: (i) large–scale data collection, (ii) supervised prediction of V.U.N. values using a neural model, and (iii) continuous optimization over the learned predictor to estimate optimal hyperparameters for arbitrary step budgets.

5.1. Large-Scale Grid Search

We begin by generating a large dataset of $(\eta, \omega, \text{distortion}, S) \to$ V.U.N. pairs. This is achieved via a broad grid search on the **Planar** dataset, covering:

- steps: from 10 to 50 for Planar,
- η: uniformly in $[0, 250]$,
- ω: uniformly in $[0, 1]$,
- the five distortions introduced in the original paper.

The goal of the grid search is *not* to find optimal values directly, but rather to generate a sufficiently diverse dataset from which a predictive model can learn how V.U.N. behaves as a function of the sampling hyperparameters.

5.2. Learning a Predictive Model for V.U.N.

To estimate V.U.N. for unseen hyperparameter combinations, we train a simple multi-layer perceptron (MLP) using the grid–search results as training data. The model receives $(\eta, \omega, \text{distortion}, S)$ as input and predicts the expected V.U.N. A more detailed explanation on the implementation is given in Appendix C.

To ensure an unbiased evaluation, we construct a **pure validation set** by running an additional random search while explicitly discarding any configuration present in the grid search. The MLP is evaluated on this held-out set to confirm that it generalizes and captures the underlying trends of the sampling process, rather than memorizing the search data.

Once validated, the model is retrained on the union of the grid–search and random–search datasets to improve accuracy before the final optimization phase.

5.3. Step-Dependent Hyperparameter Optimization

Given the learned predictor $f(\eta, \omega, \text{distortion}, S)$, we search for the optimal hyperparameters for each step count by globally optimizing the predicted V.U.N.:

$$(\eta^*, \omega^*, d^*)_S \;=\; \arg\max_{d \in \mathcal{D}} \left[\max_{\eta,\, \omega} f(\eta,\, \omega,\, d,\, S) \right].$$

Importantly, this optimization is continuous, meaning that the resulting parameters are *not* restricted to the discrete

values used during the grid search. This enables exploring the hyperparameter landscape more precisely, including step counts for which no training examples exist.

5.4. Results

Figure 17 compares the V.U.N. obtained using the fixed hyperparameters from the original paper against those predicted by our step–dependent optimization. The improvement is substantial across almost the entire range of step values. Interestingly, while the original work already reports strong performance for small step counts, our optimized parameters consistently outperform theirs, revealing additional performance gains that had remained unexplored.

Moreover, the predictor generalizes well to step values not present in the grid search, indicating that the MLP successfully learned a meaningful representation of the sampling dynamics. This suggests that the method could be applied to other datasets (e.g., QM9).

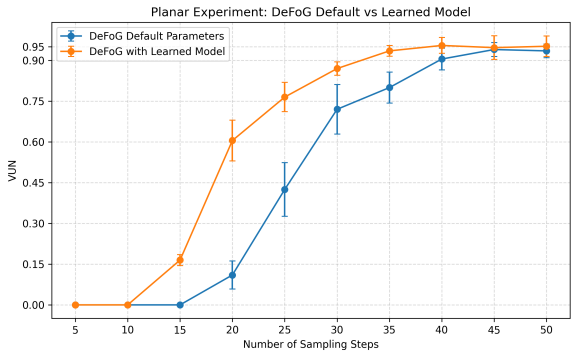


Figure 17: Comparison between V.U.N. scores obtained using the sampling hyperparameters from the original paper and those predicted by our step-dependent optimization strategy. Substantial improvements are observed across most step values.

Overall, these experiments demonstrate that sampling hyperparameters should indeed be tuned as a function of the step budget. The proposed pipeline offers a principled way to automate this process by learning a surrogate model of the sampling behavior and optimizing it continuously.

6. Conclusion

In this work, we successfully reproduced the core results of the DeFoG framework and extended it in several meaningful directions. We incorporated both structural and property-based conditional generation, allowing the model to generate graphs that respect fixed subgraphs or match target properties, capabilities that were not originally supported. We further explored novel distortion functions, showing that distortion design plays a central role in generation quality and should be considered a key modeling choice.

Additionally, we introduced a step-dependent hyperparameter optimization procedure, demonstrating that sampling hyperparameters must be tuned according to the number of denoising steps to maximize performance. Our experiments across Planar, QM9, and TLS confirm the flexibility of these extensions: from controlled structural constraints to chemically meaningful fragment conditioning and full-model retraining on more complex molecular data.

On the whole, our contributions strengthen the original DeFoG methodology and broaden its applicability, offering new tools for conditional graph and molecule generation and laying the groundwork for more expressive and controllable generative models.

References

[1] Yiming Qin, Manuel Madeira, Dorina Thanou, Pascal Frossard, *Defog: Discrete flow matching for graph generation*, arXiv preprint arXiv:2410.04263, 2024.

[2] Yaron Lipman, Ricky T.Q. Chen, Belhal Tzen, Roger B. Grosse, *Flow Matching for Generative Modeling*, International Conference on Learning Representations, 2023.

[3] Emiel Hoogeboom, Rianne van den Berg, Max Welling, *Autoregressive Flow Transformers*, International Conference on Machine Learning, 2022.

[4] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, Jure Leskovec, *GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models*, International Conference on Machine Learning, 2018.

[5] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, Pascal Frossard, *Digress: Discrete denoising diffusion for graph generation*, arXiv preprint arXiv:2209.14734, 2022.

[6] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, Jure Leskovec, *Prodigy: Enabling in-context learning over graphs*, Advances in Neural Information Processing Systems, vol. 36, pp. 16302–16317, 2023.

[7] Chence Shi, Minkai Xu, Junbo Zhu, Jian Tang, *GraphAF: Flow-based Autoregressive Generative Models for Molecular Graph Generation*, International Conference on Learning Representations, 2020.

A. Datasets

A.1. Planar

The Planar dataset consists of synthetic undirected planar graphs (graphs that can be drawn in a plane without edge crossings). The dataset contains only graph structures: nodes and edges without labels or additional attributes, making it a clean benchmark for testing structural graph generation.

A.2. QM9

The QM9 dataset is a standard benchmark for molecule generation and property prediction. It contains about 130,000 small organic molecules composed of carbon, hydrogen, oxygen, nitrogen, and fluorine. Each molecule includes a rich set of quantum-chemical properties.

We use QM9 in two variants:

- **QM9 without H:** hydrogens are removed during preprocessing, which reduces graph size and improves efficiency for generative training while keeping the overall structure.

- **QM9 with H:** hydrogens are included, producing larger graphs and richer chemical structures. This variant preserves full molecular information and is used when modeling hydrogen-dependent properties or for more chemically complete generation.

A.3. TLS

The TLS dataset contains graph representations of Tertiary Lymphoid Structures, complex immune formations that emerge in tissues during chronic inflammation, autoimmunity, or cancer. These graphs are larger and more heterogeneous than those in Planar or QM9, making them a more challenging benchmark. Each example also includes a biologically meaningful scalar property, enabling conditional generation. Because no pre-trained models existed for this dataset, we trained Defog completely from scratch and released all resulting checkpoints to support reproducibility and future research.

B. Fixed subgraphs

B.1. Planar

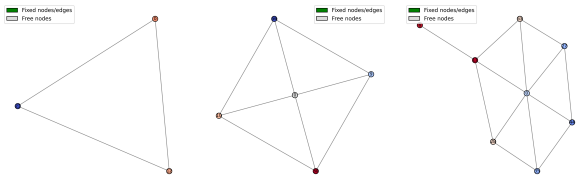
For the Planar dataset, we started from a single base graph and extracted multiple connected subgraphs with lengths ranging from 2 to 32 nodes. For each of these subgraphs, we ran the generative model 40 times, obtaining 40 complete graphs per subgraph configuration. This allowed us to study how the model behaves as we increase the number of fixed nodes.

Figure 18 illustrates this process. In panel (a), we show several of the base subgraphs extracted from the original planar graph. Panel (b) highlights how these subgraphs appear embedded within their corresponding full graphs. Finally, panel (c) presents examples of the final generated planar graphs produced by the model while conditioning on the fixed subgraphs, demonstrating how Defog completes the missing structure around the constrained region.

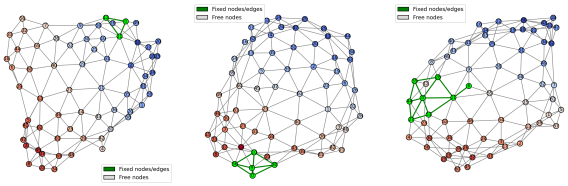
B.2. QM9

For the QM9 dataset, we selected 10 distinct molecules generated from scratch and extracted all linear sub-chains of lengths between 2 and 5 atoms. For every sub-chain, we generated 10,000 new molecules while fixing that fragment as a structural constraint. This approach lets us analyze the effect of fixing chemically meaningful fragments on the validity and stability of the generated molecules.

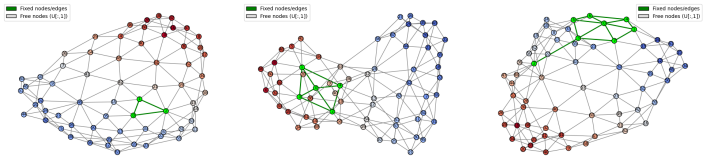
Figure 19 illustrates the full process. Panel (a) shows several of the linear atomic chains that were used as fixed fragments. In panel (b), these chains are highlighted within their corresponding original molecules, allowing us to visualize the context from which each fragment was extracted. Finally, panel (c) displays representative molecules generated while conditioning on each fixed sub-chain, demonstrating how the model reconstructs plausible chemical structures around the preserved fragment.



(a) Base subgraphs extracted from the initial planar graph.



(b) Highlighted subgraphs inside their corresponding original graphs.



(c) Example of generated planar graphs conditioned on fixed subgraphs.

Figure 18: Subgraph extraction and graph generation process for the **Planar** dataset. Each row shows how a selected subgraph is embedded and used as a constraint for new samples.

C. MLP surrogate model for V.U.N prediction

Input representation and preprocessing. Each configuration is encoded using four components: (i) the number of sampling steps; (ii) a one-hot vector representing the distortion type; (iii) the continuous parameters η and ω. All features and the regression target (V.U.N) are standardized.

Network architecture. We employ a small MLP with two hidden layers of 32 units and ReLU activations. The model contains only a few thousand parameters, which is sufficient given the structured nature of the search space. Training uses Adam with a learning rate of 10^{-3} and mean squared error. We train for 500 epochs with mini-batches of size 32.

Training and validation strategy. The training data consists of a large grid-search covering:

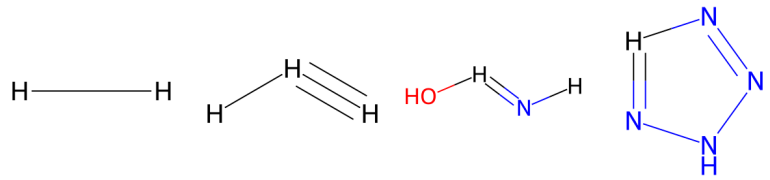
$$\texttt{num\_steps} \in [10, 50], \quad \eta \in [0, 250], \quad \omega \in [0, 1], \quad \text{five distortion types.}$$

To obtain a clean validation set, we generate an additional random-search set and remove any configurations already present in the grid-search. The model is first trained on the grid-search data and validated on the random-search data to confirm that it generalizes beyond memorizing the discrete grid. After validation, we retrain the MLP on the combined dataset to obtain the final surrogate model.

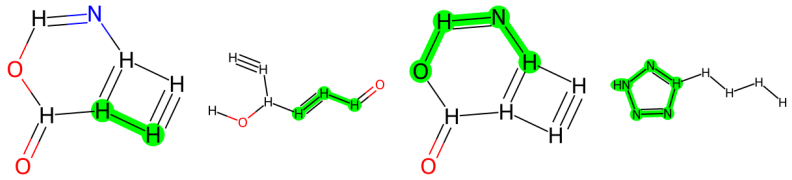
Hyperparameter optimization via the surrogate model. Given a target number of sampling steps, we aim to identify the configuration $(\eta, \omega, \text{distortion})$ that maximizes the predicted V.U.N. We treat the distortion type as a discrete option and, for each distortion, solve a continuous optimization problem over (η, ω) using differential evolution. The objective minimized by the global optimizer is the negative predicted V.U.N. Among the five optimized candidates (one for each distortion type), the configuration with the highest predicted V.U.N is selected.

C.1. Additional Checkpoints

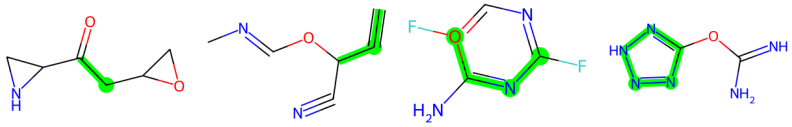
For completeness, we release the checkpoints obtained during our retraining and extensions, which are not part of the original DeFoG repository. These checkpoints correspond to the models used in our conditional–generation experiments:



(a) Base molecular chains used as fixed fragments.



(b) Highlighted sub-molecular fragments within their original molecules.



(c) Molecules generated by fixing each of the sub-chains.

Figure 19: Sub-chain extraction and generation for the **QM9** dataset. Each example shows how fixed atomic chains are preserved during generation.

QM9 (with and without hydrogen atoms) conditioned on energy, and *TLS* conditioned on the k_2 property. The checkpoints are available here.