Sampling Triangulations and Calabi-Yau Threefolds with Autoregressive GNNs

Anonymous authors
Paper under double-blind review

Abstract

We introduce 'dualGNN', an autoregressive message-passing GNN for sampling fine, regular triangulations of lattice polytopes. dualGNN operates on a generalization of the dual graph of a triangulation, with edges labeled by 'signed circuits' — combinatorial invariants from the theory of oriented matroids. We show that these circuits are necessary and sufficient to determine a triangulation's regularity from the graph, provided certain magnitude information is retained. The model is independent of the polytope's point count and invariant under its orientation-preserving symmetries ($\mathrm{SL}(d,\mathbb{Z}) \ltimes \mathbb{Z}^d$), and our masking procedure further guarantees that every rollout produces a fine triangulation (in 2D). On unseen polygons with $N_{\mathrm{pts}} \leq 40$, dualGNN is the only sampler we tested that is consistent with uniform sampling across all our diagnostics (KL divergence from uniformity, collision counts, and sample autocorrelation). The model is small ($\sim$ 92k parameters) and trains in $\sim$ 7.5 hours on a single consumer GPU. We apply dualGNN to string theory, sampling Calabi-Yau threefolds uniformly at $h^{1,1} = 86$; we also sample CYs at $h^{1,1} = 128$, observing no deviations from uniformity, but our diagnostics are weaker here. Code, training scripts, and pretrained models are available at `https://github.com/natemacfadden/dualGNN` (`pip install dualgnn`), and dualGNN is integrated into CYTools.

1 Introduction

Sampling fine, regular triangulations (FRTs) of lattice polytopes is a long-studied (over two decades Meyer (2002); Kaibel, V. and Ziegler, G.M. (2003)) discrete, combinatorial problem with rich structure. It involves

(a) large scale (the largest polygon relevant to our applications has $\geq 3.9 \times 10^{167}$ FRTs MacFadden et al. (2026)),

(b) local constraints (triangulation fineness and validity),

(c) global constraints (triangulation regularity), and

(d) nontrivial symmetries (lattice polytopes are $\mathrm{GL}(d,\mathbb{Z}) \ltimes \mathbb{Z}^d$ invariant).

There are a wide variety of approaches to this problem[1] including, but not limited to those discussed in Meyer (2002); Kaibel, V. and Ziegler, G.M. (2003); Demirtas et al. (2020); Yip et al. (2025) as well as `grow2d` from CYTools. Despite this long study and rich literature, all existing approaches tend to struggle with scale, generality, or bias; there is no efficient[2] algorithm that provides (even approximately) uniform FRT samples for general polygons. There is need for such capabilities: these triangulations define Calabi-Yau threefolds,

[1]Subsequent to our original arXiv posting, another approach Wang & Montúfar (2026) was also posted to arXiv. We do not discuss it here other than noting that it operates on 4D triangulations directly, so the redundancy concerns we later raise for CYTransformer in this section apply, and could likewise be addressed by the 2-face reduction of MacFadden (2023). I.e., restrict to flips modifying the 2-face triangulations. This can be done using existing code in CYTools.

[2]We return to Kaibel, V. and Ziegler, G.M. (2003) later.

so such samples of triangulations provide much needed statistical probes into the landscape of string theory. This was the motivation of Demirtas et al. (2020); Yip et al. (2025).

To address this sampling problem, we introduce dualGNN, an autoregressive message-passing GNN Gilmer et al. (2017) built on the theory of oriented matroids, a general combinatorial structure underlying disparate areas of discrete mathematics including that of triangulations of a polytope. Exploiting this structure, dualGNN is the only sampler we tested that is consistent with uniformity across all our diagnostics, generalizes (zero-shot!) to unseen polygons, and scales to large polygons. Our studied application of dualGNN is for sampling Calabi-Yau manifolds up to $h^{1,1} = 128$, but this problem can be viewed more generally as learning to sample certain subsets of signed circuits of the matroid, as we briefly discuss later.

1.1 Triangulations

A triangulation $\mathcal{T}$ of a lattice polytope Δ is a decomposition of Δ into simplices (obeying certain compatibility constraints De Loera et al. (2010)) with vertices taken from $\Delta \cap \mathbb{Z}^d$. See figs. 1 and 2. We say that a triangulation is 'fine' (denoted FT) if, for each $p \in \Delta \cap \mathbb{Z}^d$, there exists a simplex $\sigma \in \mathcal{T}$ such that p is a vertex of σ. In 2D, 'fine' is synonymous with 'unimodular'. Fineness can be viewed as a local constraint, verified by checking that each simplex $\sigma \in \mathcal{T}$ contains exactly $d+1$ lattice points in its support (in 2D this is an area computation).

Regularity is the more interesting constraint. A triangulation $\mathcal{T}$ is 'regular' (denoted FRT if also fine) if and only if it can be defined by the lifting procedure in algorithm 1. For 2D, this is to embed every lattice point $p_i = (x_i, y_i)$ of the polygon into $\mathbb{R}^3$ as (x_i, y_i, h_i) for some choice of $h_i \in \mathbb{R}$, construct the convex hull of these lifted points, and then project the lower envelope of this hull to $\mathbb{R}^2$ as the triangulation (see fig. 1). The height vector $h = (h_1, \ldots, h_{N_{\text{pts}}}) \in \mathbb{R}^{N_{\text{pts}}}$ is not unique: $\mathcal{T}$ is equivalently generated by any vector in the interior of the 'secondary cone' $\{h \in \mathbb{R}^{N_{\text{pts}}} : Hh > 0\}$, where H is a matrix determined by the simplices of $\mathcal{T}$. The existence of a height vector depends on the global structure of the triangulation; fig. 2 gives an example of two FRTs which become irregular after 'patching' them together.

Algorithm 1: Lifting Procedure

Input : Lattice polygon $\Delta = \text{conv}\{x_i \in \mathbf{A}\}$
Input : Height vector $h \in \mathbb{R}^{|\mathbf{A}|}$
Output: Triangulation $\mathcal{T}$ of Δ

$\tilde{\mathbf{A}} \leftarrow \{(x_i, h_i) : x_i \in \mathbf{A}\}$ `// lift points`
$\tilde{\Delta} \leftarrow \text{conv}(\tilde{\mathbf{A}})$
$F \leftarrow \texttt{LowerFacets}(\tilde{\Delta})$ `// facets with inward normal` $n_{d+1} > 0$
$\mathcal{T} \leftarrow \emptyset$
foreach *facet* $\tilde{\sigma} \in F$ **do**
  $\sigma \leftarrow \texttt{Project}(\tilde{\sigma})$ `// drop last coordinate`
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\sigma\}$
return $\mathcal{T}$

FRTs of lattice polygons arise naturally in string theory, where they provide an efficient route to enumerating Calabi-Yau threefolds. A central goal in string theory is to find Calabi-Yau threefolds (CYs) which give rise to certain desired physics (de Sitter geometry McAllister et al. (2024), standard model embeddingsCvetič et al. (2001); Aldazabal et al. (2000); Cvetič et al. (2019); Blumenhagen et al. (2009), ...). These searches cannot be done exhaustively: the largest-known collection contains up to 10^{296} MacFadden et al. (2026) CYs[3], each constructed from a certain fine, regular, 'star' triangulation (denoted FRST) of a 4D polytope Batyrev (1994). There are $\leq 10^{928}$ FRSTs Demirtas et al. (2020), making the map FRSTs $\rightarrow$ CYs many-to-one. The redundancy has a simple combinatorial description: if two FRSTs $\mathcal{T}_1$ and $\mathcal{T}_2$ of a lattice polytope Δ define the same triangulations on the 2-faces of Δ, then they generate homotopy-equivalent CYs. In our prior

[3]More pointedly, the current best algorithm for enumerating these CYs MacFadden (2023) would require iterating over at least 10^{276} items to generate this entire collection MacFadden et al. (2026).

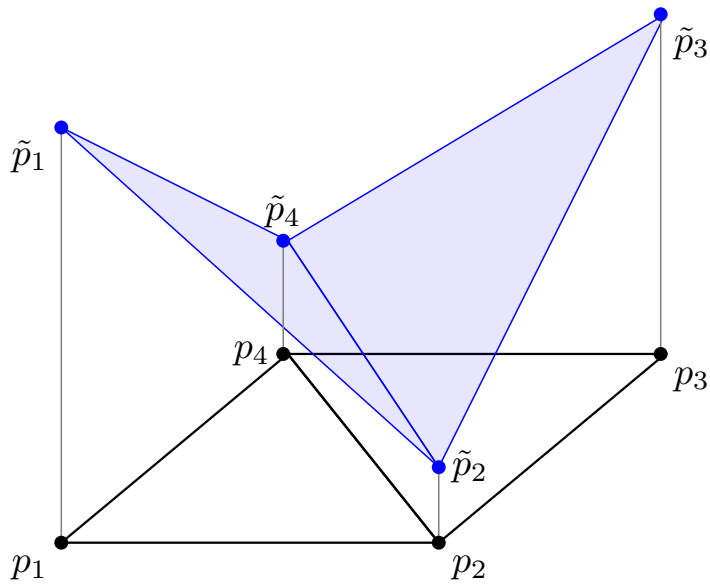

Figure 1: Diagram of the 'lifting' procedure defining regular triangulations. The points p_1, p_2, p_3, and p_4 are embedded into $\mathbb{R}^3$ and then lifted by heights $h_1 = 1.1$, $h_2 = 0.2$, $h_3 = 0.9$, and $h_4 = 0.3$. The convex hull of the lifted point configuration is a 3-simplex whose lower faces are plotted in blue. Projecting out the lifted coordinate generates the regular triangulation plotted in black. Figure modified from ref. MacFadden (2023).

work MacFadden (2023) we developed an algorithm that turns this redundancy into a constructive tool: given a set of triangulated 2-faces, one can directly construct a compatible FRST if one exists, or obtain a certificate that no such FRST exists. This reduces CY enumeration to the generation of FRTs of polygons (2D), sidestepping the exponentially-redundant FRST space and motivating the focus on polygons in this work.

1.2 The dualGNN Representation

To sample FRTs of polytopes (particularly, polygons), dualGNN operates on a graph whose nodes are candidate simplices and whose edges connect pairs of simplices that share a facet with no overlapping interior. See right side of fig. 3. As discussed in section 2, each edge carries a fixed feature vector λ, called a 'signed circuit', which encodes a linear dependency among the lattice points in the adjacent simplices $\sigma \cup \sigma'$. Writing $\mathbf{A}_{\sigma\cup\sigma'}$ for the matrix whose columns are these points, λ satisfies

$$\mathbf{A}_{\sigma\cup\sigma'}\,\lambda = 0 \quad \text{and} \quad \sum_i \lambda_i = 0. \tag{1.1}$$

We will loosely call λ itself a 'circuit' despite it having strictly more information[4] than an actual signed circuit from oriented matroidBjörner et al. (1999) theory. Of note:

(a) Circuits encode the oriented matroid of the polytope Δ, which determines the combinatorial structure of all triangulations of Δ.

(b) For a regular triangulation, these vectors λ are exactly the hyperplane normals H defining its secondary cone, so edges directly encode the regularity constraints.

[4]A signed circuit would more honestly correspond to the decomposition of λ into positive and negative indices, without magnitudes. We carry the relative magnitude of components since those are crucial, e.g., to determine the regularity of a triangulation — see De Loera et al. (2010) section 7.1.1.

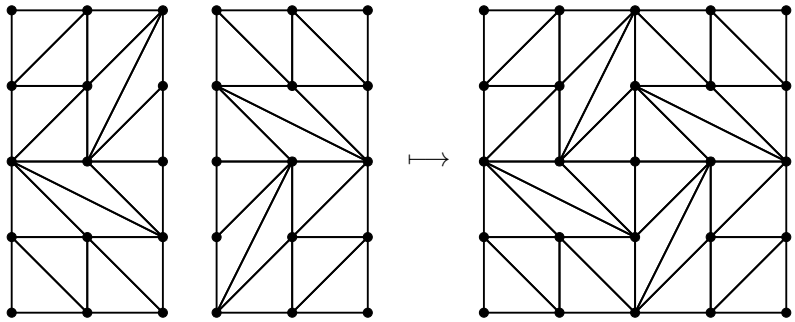

Figure 2: Two fine regular triangulations of $[0,2] \times [0,4]$ being 'patched' to a single triangulation of $[0,4]^2$. The two triangulations on the left are both regular while the triangulation on the right is irregular. This example was originally found by Francisco Santos and appears in ref. Kaibel, V. and Ziegler, G.M. (2003). Figure modified from MacFadden et al. (2026).

(c) As we demonstrate in section 2, circuits are invariant under the symmetries of the polytope (translation and unimodular maps). dualGNN is therefore invariant under the orientation-preserving subgroup[5] $\mathrm{SL}(2,\mathbb{Z}) \ltimes \mathbb{Z}^2$ of these symmetries.

Our circuit-based network dualGNN is autoregressive in the style of a Pointer Network Vinyals et al. (2015), predicting probability distributions over its inputs instead of a fixed vocabulary. More explicitly, at each step, K rounds of message-passing on the graph produce a probability distribution over candidate simplices, a simplex σ is sampled, and then other simplices that overlap σ are masked. The initial construction of the graph enforces fineness while our masking enforces validity. This implies, for a 2D polytope, every rollout will produce a valid, fine triangulation. We utilize both supervised learning and RL. We need RL since the cross-entropy objective in our supervised learning teaches conditional probabilities according to our *estimated* targets (when we are bootstrapping a pool of triangulations), so bias in the training pool propagates to the sampler. We correct this by fine-tuning with REINFORCE Williams (1992) under an entropy-maximizing reward, directly optimizing the rollout's distribution toward uniformity (section 2.2.3).

Empirically, dualGNN matches a uniform sampler more closely than any baseline we tested across our diagnostics. Uniformity is surprisingly subtle to measure here, primarily because some polygons we study have up to 10^{21} FRTs while our computational budget limits us to $\lesssim 10^7$ samples. When the FRT count is $\lesssim 10^6$, we use diagnostics like KL divergence to the uniform (flat) distribution; for larger FRT counts, KL divergence becomes less informative due to low (but often nonzero) collision counts. In such cases, just comparing the empirical collision count to theoretical predictions (the birthday problem in probability theory) proves more useful. Additionally, for polygons of all sizes, we measure the autocorrelation between consecutive samples, analogous to standard tests for pseudorandom number generators Knuth (1997). Our single 92k-parameter model, trained in ~ 7.5 hours on an NVIDIA RTX 5060 Ti, generalizes to held-out polygons with N_{pts} as large as 40, the maximum we tested. This model even runs on common laptops like an M1 MacBook Pro.

1.3 Prior Work

The problem of sampling FRTs has an extensive literature. A standard approach is a flip-walk on the space of triangulations, which converges to the uniform distribution but whose mixing is poorly understood Kaibel, V. and Ziegler, G.M. (2003); Felsner & Wernisch (1997). The authors of Kaibel, V. and Ziegler, G.M. (2003) also provide the first (to our knowledge) autoregressive sampler of fine triangulations, which is *provably uniform*, but it scales exponentially on the 'width' of their polygons (for $m \times n$ lattice rectangles, their algorithm requires at most $10^m(n+1)^{m+1}$ arithmetic operations). This could be improved by the methods of Orevkov (2022) but the exponential scaling would persist. Other non-learned methods are discussed in Demirtas et al. (2020), including `random_triangulations_fast` and `random_triangulations_fair`, both

[5]The orientation-reversing case is addressed in section 2.

Figure 3: Left: two simplices share a facet but in an invalid way. The simplices have a solid (2D) intersection. Right: two simplices share a facet in a proper way. Their intersection is only on the facet $\mathrm{conv}(\{e_k, e_\ell\})$. Edges are drawn between pairs of simplices with such proper intersections only.

implemented in CYToolsDemirtas et al. (2022). Additionally, with the publishing of MacFadden (2023), we released an autoregressive simplex-based method `grow2d` with CYTools (commit `9f5ef24`). We also introduce an algorithm `pushing` in this work, inspired by TOPCOM Rambau (2002).

There are also a collection of relevant learned methods. Broadly, there is work on sampling from combinatorial spaces, including discrete normalizing flows Tran et al. (2019); Hoogeboom et al. (2019) and GFlowNets Bengio et al. (2021) among others. GFlowNets are especially relevant: with reward $R(\mathcal{T}) = 1$ for regular $\mathcal{T}$ (and 0 otherwise), a GFlowNet converges to the uniform distribution over FRTs; we discuss the relation to our training objective in section 2.2.3. More tailored to FRT sampling, there is CYTransformer Yip et al. (2025), which is an encoder-decoder that autoregressively generates simplices conditioned on encoded polytope vertices. Finally, a distinct but complementary line of learned methods optimizes functions over CYs via reinforcement learning and genetic algorithmsBerglund et al. (2025); MacFadden et al. (2025). These works are complementary in that they consume FRTs of polygons, rather than generating them.

In particular, we will make heavy use of the classical baselines `random_triangulations_fast` (which we abbreviate to `fast`)[6], `pushing`, and `grow2d`. We also compare against random walks in flip space (`flip_walk`; algorithm #1 of Demirtas et al. (2020)). We notably exclude `random_triangulations_fair` despite it being designed for uniformity because this algorithm adds height-modification steps to `flip_walk` which make the sampler prohibitively slow for our work.

The most-similar previous work is CYTransformer Yip et al. (2025). Like dualGNN, it seeks to autoregressively generate fine, regular triangulations of lattice polytopes for applications in CY generation. Yip et al. (2025) utilizes an encoder-decoder transformer to achieve strong performance in generating regular triangulations (something we partially recreate, albeit for decoder-only transformers, in appendix C). dualGNN shares this learned-autoregressive framing but differs in architecture and intermediate representation, as described below.

(a) Yip et al. (2025) trained their CYTransformer encoder-decoder model for each N_{vert} of interest (analogous to N_{pts} in our work), with token vocabulary of size $\binom{N_{\text{vert}}-1}{4}$; we train a *single* dualGNN model to operate on general N_{pts} since there is no fixed vocabulary,

(b) CYTransformer utilizes a coordinate-based representation of the polytope, which is not symmetry invariant. They address point relabeling via data augmentation but did not address other symmetries; dualGNN benefits Bronstein et al. (2021) from having symmetries built into the architecture, and

(c) CYTransformer samples CYs via triangulations of 4D polytopes; we instead use the 2-face decomposition of MacFadden (2023) to more directly sample the (potentially) homotopy-inequivalent ones.

The first two differences allow dualGNN, unlike CYTransformer, to generalize zero-shot across polytopes of different sizes, shapes, and N_{pts}. The third difference enables us to sample CYs at significantly higher

[6]We note that `random_triangulations_fast` was not advertised by Demirtas et al. (2020) as a uniform sampler. We include it as a biased reference.

$h^{1,1}$: by generating triangulated 2-face data directly, we avoid the exponential FRST redundancy ($N_{\text{FRST}} \lesssim 10^{1.91h^{1,1}-5.31}$ compared to $N_{\text{CY}} \lesssim 10^{0.9h^{1,1}-15.45}$ Demirtas et al. (2020)) that a 4D sampler must contend with. This enables dualGNN to sample up to $h^{1,1} = 128$ compared to their $h^{1,1} \leq 10$. dualGNN is also $\sim 1300\times$ smaller and significantly cheaper to train (7.5 hours on an RTX 5060 Ti compared to Yip et al. (2025)'s few days per $(h^{1,1}, N_{\text{vert}})$ on 8 NVIDIA V100 GPUs). We stress that CYTransformer could likely be made smaller, quicker to train, and operable at higher $h^{1,1}$ if it adopted the 2-face decomposition of MacFadden (2023) (as we do here), though these do not address its difficulty in generalizing across polytopes. A direct head-to-head would require this retraining, which is outside our budget given CYTransformer's currently unavailable weights and source; we therefore compare only via their figure-4 diagnostic in section 3.1.

1.4 Contributions, Paper Structure

In summary, our contributions are:

(a) A symmetry-respecting encoding for triangulation problems. We label the edges of a generalized dual graph with magnitude-bearing signed circuits from the polytope's oriented matroid, yielding an architecture invariant under $\text{SL}(d, \mathbb{Z}) \ltimes \mathbb{Z}^d$ and independent of N_{pts} (section 2). Sign-only circuits are provably insufficient for regularity, while our magnitude-bearing 'circuit' features coincide with the secondary-cone normals and thus the encoding exposes regularity (validated empirically in section 2.1).

(b) Constraint satisfaction by construction. Graph construction enforces fineness and autoregressive masking enforces validity, so in 2D every rollout is a valid fine triangulation (section 2.2).

(c) Size generalization and zero-shot transfer. A single ∼92k-parameter pointer-network-style sampler, trained in hours on one consumer GPU, transfers zero-shot across polygons of different sizes and shapes — even when trained on just a single polygon (sections 2.2.1 and 2.2.2).

(d) A bias-inheritance fix. Teacher-forced training on bootstrapped pools inherits their bias; entropy-reward REINFORCE fine-tuning corrects it, yielding the only sampler we tested that is consistent with uniformity across all diagnostics (section 2.2.3).

(e) Application. Combined with the 2-face reduction of MacFadden (2023), we sample Calabi-Yau threefolds uniformly at $h^{1,1} = 86$ and observe no deviations from uniformity at $h^{1,1} = 128$ (where our diagnostics are weaker), an order of magnitude larger than prior learned samplers (section 3).

The rest of the paper is as follows. Section 2 introduces dualGNN in detail. Section 2.1 evaluates it as a regularity classifier; section 2.2.1 as an FRT sampler trained on a single polygon; section 2.2.2 demonstrates zero-shot transfer between polygons; and section 2.2.3 as a general-purpose sampler trained across polygon configurations. Section 3 then applies the general-purpose sampler to Calabi-Yau enumeration up to $h^{1,1} = 128$. Section 4 discusses scope and scalability; section 5 concludes.

2 dualGNN

We first introduce a simplified variant of dualGNN; the more complete model will be discussed in section 2.2. Consider encoding a triangulation $\mathcal{T}$ of Δ by its dual graph $G_{\mathcal{T}}$. That is, draw a node for each simplex of $\mathcal{T}$ and edges between any two nodes whose corresponding simplices share a facet (call such simplices 'adjacent'). Such a graph encodes the triangulation as a simplicial complex, but it obscures some geometric data. For example, each pair of simplices in fig. 4 would correspond to an edge in $G_{\mathcal{T}}$, but they play different roles in a triangulation as we now describe.

The combinatorial structure of triangulations is often described in the language of oriented matroidsDe Loera et al. (2010). In this language, one characterizes the transformations a triangulation can take ('flips'), the validity of a triangulation, the total number of triangulations of Δ, etc. in terms of certain objects called[7]

[7]Other equivalent objects like cocircuits and chirotopes also equivalently characterize the triangulationDe Loera et al. (2010). Circuits are most-directly useful for this work, so we use them.

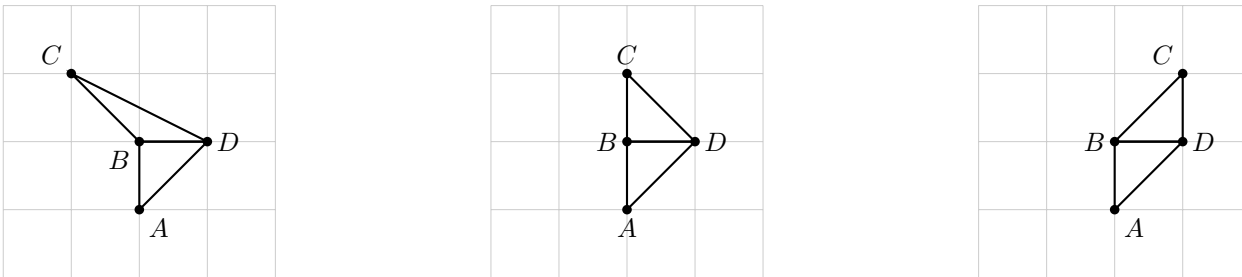

Figure 4: Pairs of adjacent simplices ABD and BCD. Each pair corresponds to an edge in a dual graph, but these pairs correspond to circuits playing different roles in the oriented matroid so they must be distinguished.

'signed circuits' (often just called 'circuits'). If we organize the lattice points of Δ as the columns of a matrix $\mathbf{A}$, the circuits correspond to certain minimal (in terms of the count of nonzero elements) vectors λ satisfying

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{1} \end{bmatrix} \lambda = 0. \tag{2.1}$$

Circuits are sparse, with $\leq d+2$ nonzero elements (for a d-dimensional polytope). Additionally, only a small subset of circuits are relevant to a triangulation $\mathcal{T}$: every flip of $\mathcal{T}$ is characterized by a circuit λ such that $\text{nonzero}(\lambda) \subseteq \sigma \cup \sigma'$ for σ, σ' adjacent in $\mathcal{T}$.

Return to the pairs in fig. 4. Each pair corresponds to an edge in $G_{\mathcal{T}}$ but such edges describe different types of flips/circuits. The leftmost two pairs describe flips in which a point is deleted (albeit in different ways) while the rightmost pair describes a diagonal flip. These are fundamentally different transformations which must be distinguished for a faithful representation of the oriented matroid, but $G_{\mathcal{T}}$ is blind to their differences. The primary idea of dualGNN is to directly inject this circuit information into $G_{\mathcal{T}}$ as a fixed feature of each edge, appending this feature to any message passed through it. The circuit data respects the $\mathrm{GL}(d, \mathbb{Z}) \ltimes \mathbb{Z}^d$ symmetries of the polytope:

$$\left(\mathbf{A} + \begin{bmatrix} r & \cdots & r \end{bmatrix}\right) \lambda = 0 \iff \mathbf{A}\lambda = 0 \iff (\mathbf{U}\mathbf{A})\lambda = 0, \tag{2.2}$$

for unimodular $\mathbf{U}$. In the first equality, we used that $\sum_i \lambda_i = 0$. The same circuit data, and hence the same edge-labeled graph $G_{\mathcal{T}}$, is therefore produced regardless of how the polytope is translated or unimodularly transformed. We stress: our above discussion on circuits applies to any dimension. The directional encoding introduced below for $d = 2$ reduces this to $\mathrm{SL}(2, \mathbb{Z}) \ltimes \mathbb{Z}^2$ by sacrificing orientation-reversal invariance.

One technical subtlety: λ above is not technically a circuit, but is related to one. A circuit is more precisely a decomposition of λ into the indices with positive and negative coefficients (defining a 'minimally dependent subconfiguration of Δ'), with magnitude information discarded. We retain λ itself, including the magnitudes, because the relative magnitudes are necessary for determining regularity (see the example in §7.1.1 of De Loera et al. (2010)). That example also validates the necessity of our encoding: it exhibits two triangulations with the same dual graph $G_{\mathcal{T}}$ and the same magnitude-stripped circuits, but different regularity (one regular, one irregular). Without λ-features, no function of $G_{\mathcal{T}}$ alone could distinguish them — even augmenting $G_{\mathcal{T}}$ with true circuits (sign patterns of λ) is insufficient; only the full magnitude-bearing λ exposes a regularity signal.

The only intricacy is in how one actually assigns a circuit λ to an edge. The core data of the circuit is the map from vertex i to its coefficient λ_i. We focus on 2D problems for which we propose the following encoding:[8] to the edge from node a to node b, assign a 4D vector $\mathbf{C}_{ab} = (\lambda_{v_i}, \lambda_{v_j}, \lambda_{e_k}, \lambda_{e_\ell})$ in which (see fig. 3)

(a) v_i is the vertex unique to node-a (the sender),

(b) v_j is the vertex unique to node-b (the receiver),

[8]This is obviously not the only possible way to encode the circuit to an edge.

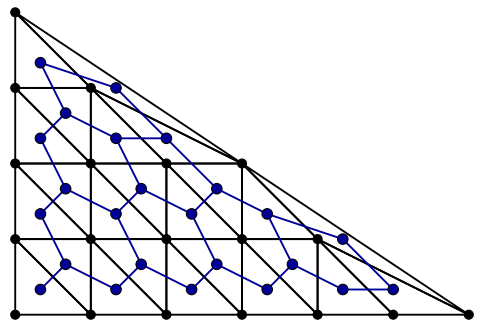

Figure 5: A fine triangulation $\mathcal{T}$ of the polytope $\Delta = \text{conv}(\{(0,0),(0,4),(6,0)\})$ (in black) as well as its dual graph $G_{\mathcal{T}}$ (in blue). This polygon has $408,826$ triangulations of which all but $3,120$ are regular.

(c) e_k is the vertex 'to the left', and

(d) e_ℓ is the vertex 'to the right'.

By 'left' and 'right', we mean: compute the signed areas

$$\text{area}_k = \det \begin{bmatrix} (v_i - e_k)_1 & (v_i - e_k)_2 \\ (v_j - e_k)_1 & (v_j - e_k)_2 \end{bmatrix} \qquad \text{area}_\ell = \det \begin{bmatrix} (v_i - e_\ell)_1 & (v_i - e_\ell)_2 \\ (v_j - e_\ell)_1 & (v_j - e_\ell)_2 \end{bmatrix} \tag{2.3}$$

and enforce $\text{area}_k > \text{area}_\ell$. Observe: these edges are directed $\mathbf{C}_{ab} \neq \mathbf{C}_{ba}$. This ordering is chosen because it represents the circuit in a locally-meaningful way and it is invariant under any point relabeling, but it costs invariance under orientation-reversing transformations ($\det(\mathbf{U}) = -1$). That is, the dualGNN we are now constructing is only invariant under $\text{SL}(2,\mathbb{Z}) \ltimes \mathbb{Z}^2$.

Since they encode the oriented matroid (the combinatorics of a triangulation), circuits are attractive for describing a triangulation. Such an architecture is especially interesting for applications to regular triangulations. For a regular triangulation $\mathcal{T}$, the hyperplane normals of the secondary cone (i.e., rows of H) are exactly these dependencies λ (with 0-coefficients for other points). That is, the circuits *are* the constraints on height space, defining the secondary cone. Such an encoding, $G_{\mathcal{T}}$ with edges labeled by signed circuits, then also encodes the secondary cone of $\mathcal{T}$. Since a triangulation is regular if and only if its secondary cone is full-dimensional, this encoding directly exposes regularity to the model.

2.1 Regularity Classifier

Before applying dualGNN as a sampler, we first verify that message-passing can read out the regularity signal in principle. We do this by configuring dualGNN as a binary classifier on complete triangulations: given $\mathcal{T}$, predict whether it is regular. Note that this task is trivial when H is known since an LP can check the feasibility of $Hh > 0$ directly. In this way, this task is a diagnostic, not an end application: it simply verifies that the network can convert circuit features into a regularity signal. This will not prove that the dualGNN autoregressive sampler can do so mid-rollout. We return to this gap in sections 2.2 and 4.

We configure the dualGNN with 32-dimensional feature vectors f_a on each node n_a (initialized to 0) and $K = 16$ message-passing rounds. Message-passing operates by:

(a) normalizing the $D = 32$-dimensional feature vector f_a,

(b) forming the message $[f_a, \mathbf{C}_{ab}]$ to node n_b,

(c) simultaneously sending all such messages (one from each node n_a to each neighbor n_b), having node n_b aggregate all incoming messages with sum, min, and max (aggregate length $3(D+4)$),

(d) concatenating the receiving node's own (normalized) feature vector f_b (total length $3(D+4)+D$), and then

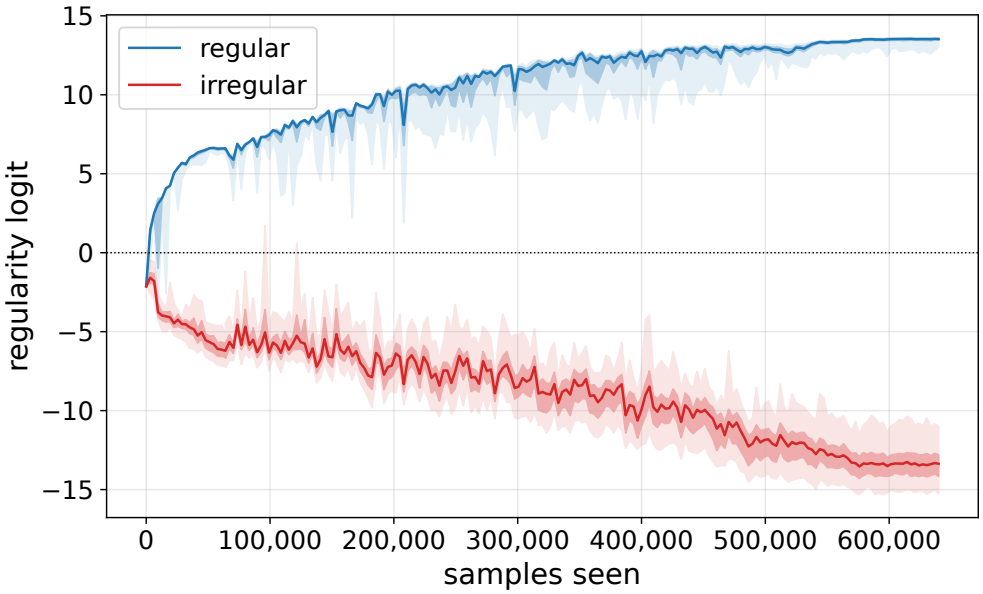

Figure 6: Performance of the regularity classifier trained and validated on the polygon in fig. 5. This evaluation is on validation data of said polygon (unseen during training; 15% of the total regular/irregular triangulation pool), with each data point representing the classification of 256 random samples from each validation pool. Note that small polygons have few irregular triangulations, that in fig. 5 is no exception, so the total irregular pool here is somewhat small $(3,120)$. The classifier first achieved $\geq 99\%$ accuracy on both regular and irregular triangulations by $\sim 35,000$ samples seen.

(e) running the combined vector through an MLP (linear layer mapping to dimension-D; GELU; linear layer mapping to D-dimension),

after which the result is added to the node's current f_b. After K rounds of message-passing, an output signal is achieved by first taking the softmin[9] over nodes and then projecting the resultant vector to a scalar which we call the 'regularity logit'. We train this model via BCE on classified data ($y = 1$ for regular $\mathcal{T}$; $y = 0$ for irregular) with loss

$$\text{loss} = -[y \log(\text{sigm}(\text{regularity logit})) + (1 - y) \log(1 - \text{sigm}(\text{regularity logit}))]. \tag{2.4}$$

For all supervised training in this paper, we use the AdamW optimizer Loshchilov & Hutter (2019) ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01) with learning rate 1×10^{-3}, batch size 16 or 32, and gradient clipping by magnitude 1. Hyperparameters were chosen by light manual search; we did not perform a systematic sweep. We did explore larger D and K values (here and for later sampling purposes), but they consistently yielded worse results despite the increased parameter count.

For this simple test, we study the polygon in fig. 5 due to its relatively high fraction of fine irregular triangulations $(3,120/408,826)$ despite the small count $408,826$ of fine triangulations. Of these triangulations, we split the regular and irregular ones each into training/validation pools by assigning 15% of each regularity class to validation. During training, triangulations are sampled from the training pool with equal probability of regular vs. irregular. We assess the performance of dualGNN by testing it intermittently on the validation pool during training (see fig. 6). dualGNN quickly learns regularity, achieving $> 99\%$ accuracy in both regular and irregular classification by $35,000$ samples seen.

[9]The softmin is motivated by an LP-feasibility view of regularity, originally explored in a dual variant of this architecture. One can construct a graph whose nodes are lattice points and edges are the edges of a triangulation (inspired by the message-passing simplicial network designBodnar et al. (2021)). This graph naturally, even without learning, propagates constraints on vertex heights by message passing between nodes mediated through circuits. For such an architecture, the regularity signal is whether the upper bound on the heights is larger than the lower bound for all nodes; one wants to take the min over upper − lower. Despite the graph used by dualGNN being dual to this variant, it still inspires the softmin choice.

2.2 As an Autoregressive Triangulation Sampler

Our focus is on using dualGNN as an autoregressive sampler of FRTs. For this task, we must generalize dualGNN. First, the graph: instead of the graph $G_{\mathcal{T}}$ dual to some triangulation $\mathcal{T}$, we use the minimal graph[10] G which contains all $G_{\mathcal{T}}$ for all fine $\mathcal{T}$. This is computed by collecting, out of the $\binom{N_{\text{pts}}}{3}$ possible simplices (in 2D), only those containing exactly 3 lattice points. For a matroid-flavored definition, G can be defined as a certain subset (with signs) of all circuits in the matroid. A circuit λ defines a unimodular simplex[11] $\{j \neq i : \lambda_j \neq 0\}$ for each i such that $\lambda_i > 0$. This simplex actually has area $\lambda_i \left|\det A_{\{j\neq i \text{ s.t. } \lambda_j \neq 0\}}\right|$, showing that $\lambda_i = 1$ for circuits relevant to unimodular triangulations. This restriction, viewed either way, guarantees fineness of any triangulation built from these nodes.

We then draw an edge $\mathbf{C}_{ab}$ between any two nodes n_a and n_b whose simplices σ_a, σ_b properly share a facet $f = \sigma_a \cap \sigma_b$ (right side of fig. 3). The resulting graph is modest: despite the polygon $[0,4]^2$ in fig. 2 having $736,983,568$ fine triangulationsKaibel, V. and Ziegler, G.M. (2003), G consists of only 320 nodes (out of the $\binom{25}{3} = 2300$ possible ones). Even the most extreme polygon occurring in our applications in string theory, $\text{conv}(\{(0,0),(0,7),(84,0)\})$, with up to 2.0×10^{180} FRTs MacFadden et al. (2026), defines a graph with only $69,416$ nodes.

The autoregressive sampler processes G in stages. At stage n, a set of n nodes will have been selected from G; these nodes correspond to n simplices forming a partial triangulation $\mathcal{T}_n$ being extended toward a complete triangulation. The model then predicts, for each outstanding node corresponding to a simplex σ, the fraction of complete triangulations (optionally, restricted by regularity) extending $\mathcal{T}_n$ that also include σ. This is done by projecting each node's feature vector to a scalar and then taking the softmax over such scalars (overriding the logits of already placed or masked simplices to $-\infty$). A simplex σ is then sampled according to these weights and all other simplices σ' with $\dim(\sigma \cap \sigma') = d$ are masked out since they cannot occur in any valid extensions. In 2D, this guarantees that the network always generates an FT: any uncovered region admits a triangulation using only its existing lattice points, so a legal next simplex always exists. There is no guarantee in higher dimensions (see, e.g., the Schönhardt polyhedron Schönhardt (1928)). Regardless, this is significantly better than the corresponding situation for a transformer, which has *no* guarantees about validity/fineness. For example, even when trained on many polygons, the transformers in appendix C struggled to generalize across polygons in our experiments.

For the network to be able to do this selection, we must modify the message-passing. Specifically, we inject, between the aggregation and the MLP, the vector $s_b = (\text{placed}, \text{legal})$ into any message sent to node n_b. This makes the total message length that the MLP sees $3(D+4) + D + 2$. Here, 'placed' and 'legal' are binary variables indicating whether node n_b has already been placed and whether it can be chosen in future rounds, respectively. We also, in contrast to section 2.1, use this s to initialize the node features as $f_a = \text{MLP}(s_a)$.

Conceptually, dualGNN's inference loop is a learned generalization of `grow2d` (see appendix B): both pick simplices one at a time and mask out incompatible candidates. `grow2d` makes uniform random choices among legal continuations that share a facet; dualGNN learns the conditional probabilities of each continuation from the circuit features. dualGNN differs by not limiting attention to new simplices sharing a facet, but that is surface-level.

Since the model operates autoregressively on partial triangulations, we train it on random prefixes (subsets of complete triangulations). Specifically, each training step selects a positive integer k (uniformly chosen) and then selects a subset of k simplices from a triangulation $\mathcal{T}$ drawn from the training pool. For each prefix, the network performs a forward-pass, predicting simplex probabilities, and then it is given loss equal to the cumulative cross-entropy compared to the ground-truth probabilities. If the ground truth is not known (common for large polygons), one estimates conditional probabilities from a bootstrap pool of triangulations sampled intermittently during training (similar to the strategy in Yip et al. (2025)). Bias in this estimate can then propagate to the trained model; we address this in section 2.2.3 by fine-tuning with REINFORCE.

[10] As an aside, we note that G is not always connected: consider $[0,1]^2$.

[11] Some care needs to be taken with homogenization. Additionally for so-called $(2,1)$-deletion circuits.

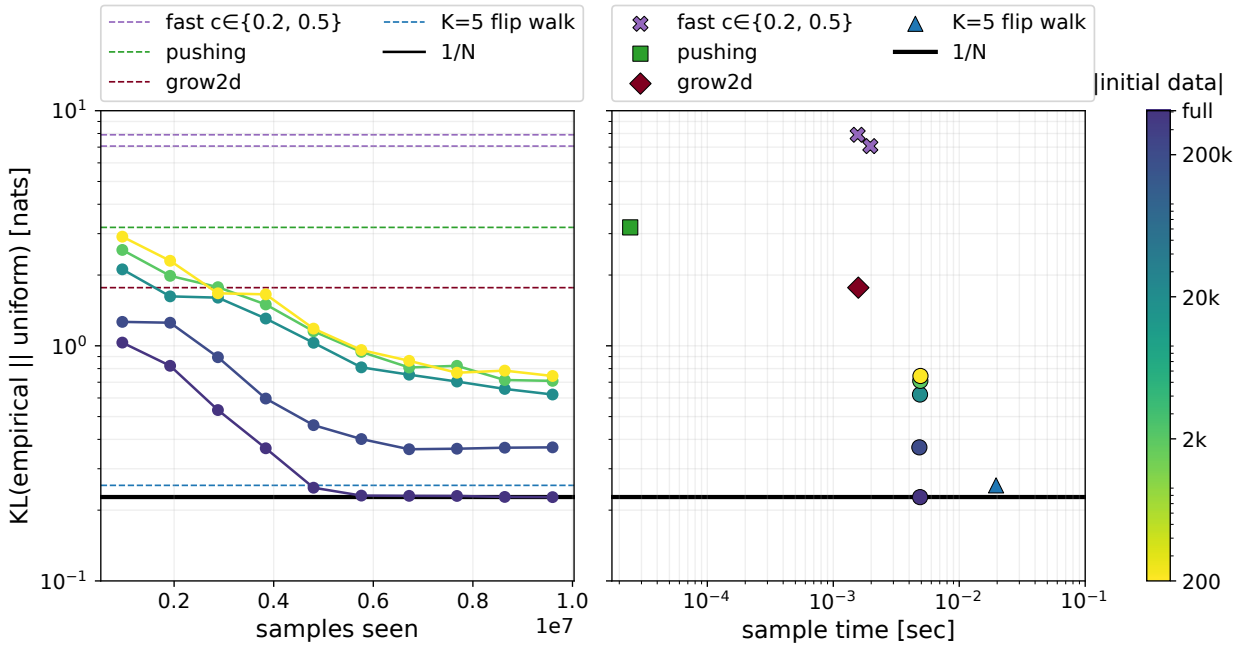

Figure 7: Performance of the autoregressive dualGNN on the polygon in fig. 5. Left: the uniformity of 10^6 samples (total FRT count is $405,706$) drawn from checkpoints mid training for varying levels of initial training data (from a full distribution down to 200 triangulations). dualGNN with access to all triangulations as training data generates more uniform (lower KL) samples than all other methods by $\sim 5 \times 10^6$ samples seen. The dualGNN variants with reduced training data beat all methods other than `flip_walk`. Right: the KL divergence of all sampling methods versus the average time to draw a sample. `pushing` is, by far, the quickest method since it does not need to check regularity of its outputs, but it is very biased and unable to generate some regular triangulations. dualGNN is approximately four times as quick as `flip_walk`.

During inference, one samples a simplex σ according to the predicted weights, masks out σ' for which $\dim(\sigma \cap \sigma') = 2$, and then repeats. This guarantees[12] that the network always generates an FT (but not necessarily regular). See appendix A for a visualization. We note that while the circuit encoding is sufficient to expose regularity (section 2.1), the autoregressive sampler does not always succeed in generating a regular triangulation: some fraction of rollouts produce irregular triangulations. For our later inference on the polygons in fig. 15, this irregular sample rate is $\sim 0.6\%$ on average, but this is highly dependent on polygon. This could suggest that our training signal restricting to regularity was insufficient. Independently, since similar training approaches worked for the transformers studied in appendix C (see fig. 27), the gap may also reflect an architectural limitation. Moving to a Graphormer-style architecture Ying et al. (2021), which gives a more global simplex signal, or even a GFlowNet could resolve this.

2.2.1 Single Polygon

We begin by training and testing dualGNN on the polygon from fig. 5 since we can fully enumerate and classify all of its FTs ($405,706$ regular triangulations and $3,120$ irregular ones). As points of comparison, we use four classical samplers: `fast` Demirtas et al. (2020), a Markov-chain sampler `flip_walk` (algorithm #1 of Demirtas et al. (2020); this is closely related to the algorithm described in section 4.1 of Kaibel, V. and Ziegler, G.M. (2003)), and two new methods `pushing` (inspired by methods in TOPCOM Rambau (2002)) and `grow2d`. These are discussed in detail in appendix B.

dualGNN outperforms the baselines we tested, achieving a lower KL divergence (compared to a uniform/flat distribution) at 10^6 samples. See fig. 7 (for those unfamiliar with KL divergence, see a visualization in

[12] Again, this fact is unique to 2D.

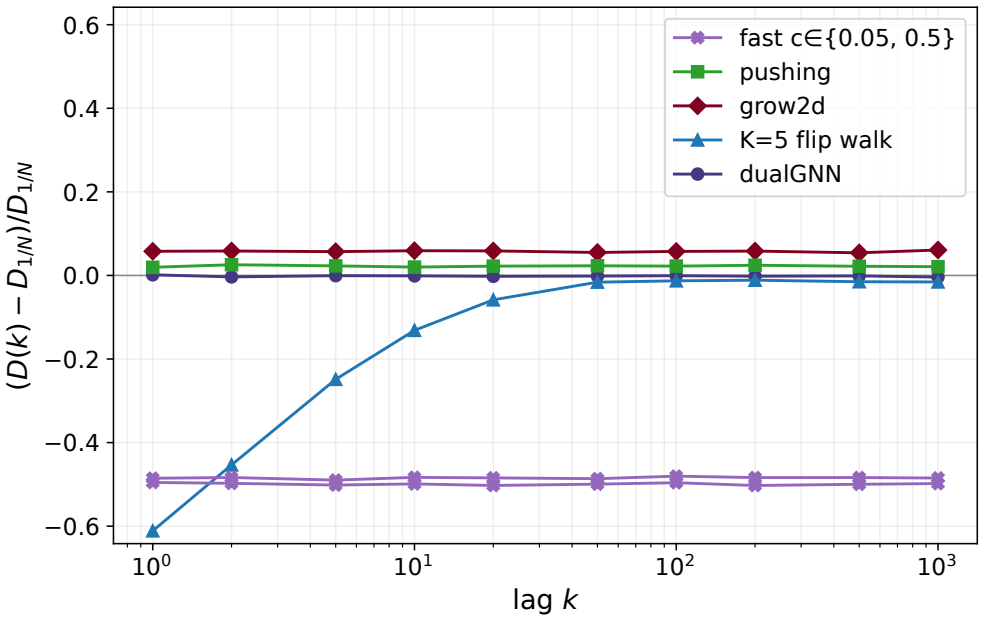

Figure 8: For the data in fig. 7, the autocorrelation between samples. Distance between samples is measured via $\mathrm{dist}(\mathcal{T}_i, \mathcal{T}_j) \geq \frac{1}{4}\left|(\mathcal{T}_i \cup \mathcal{T}_j) \setminus (\mathcal{T}_i \cap \mathcal{T}_j)\right|$; autocorrelation via $(\mathrm{dist}(\mathcal{T}_i, \mathcal{T}_{i+k}) - \mathrm{dist}_{1/N})/\mathrm{dist}_{1/N}$. Here, $\mathrm{dist}_{1/N}$ is the empirically-measured distance between samples of a uniform sampler. This is analogous to how correlation is diagnosed in random number generators Knuth (1997). All methods other than `flip_walk` show constant distances, with only dualGNN having a value agreeing with a uniform sampler (0). `flip_walk`, on the other hand, shows strong correlation for samples up to ~ 20 draws apart.

figs. 25 and 26 albeit for a different set of data). In fact, even with significantly reduced initial data (down to < 200 triangulations total, $< 0.5\%$ of all triangulations), dualGNN outperforms all methods other than `flip_walk` which also showed strong performance. For dualGNN variants with reduced training data, we use the bootstrap procedure described in section 2.2: every 500 training steps, the current model generates 100 candidate triangulations, and newly found valid triangulations are added to the training pool. This matters because real applications often involve polygons with astronomical counts of FTs (we study in section 2.2.3 polygons with up to 10^{21} FTs), for which one necessarily begins with a very small subset of all triangulations. While this bootstrapping performance is impressive, it is not unique to dualGNN: Yip et al. (2025) observed similar behavior with their transformer model and we recreate similar behavior with a transformer in appendix C.

KL divergence is not the entire story. When assessing pseudorandom number generators, one typically also studies the correlation between samplesKnuth (1997). We do the same here for our triangulation samplers. Explicitly, we use the symmetric difference between triangulations as a lower bound on flip distance[13] $\mathrm{dist}(\mathcal{T}_i, \mathcal{T}_{i+k}) \geq \frac{1}{4}\left|(\mathcal{T}_i \cup \mathcal{T}_{i+k}) \setminus (\mathcal{T}_i \cap \mathcal{T}_{i+k})\right|$ for samples i and $i+k$. Ideally, on average, this flip distance should match that of a true $1/N$ sampler, so we plot the normalized difference from this baseline in fig. 8. The methods `pushing` and `grow2d` show constant distances, as expected from a uniform sampler, but they show unexpectedly *high* flip distances, indicating these samplers under-sample nearby triangulations rather than over-sampling them. `fast`, on the other hand, shows constant distances consistently below the uniform expectation, indicating the sampler over-samples nearby triangulations. dualGNN is the only sampler whose flip distances are consistent with a true uniform distribution — constant and at the expected value compared to a uniform sampler. Finally, `flip_walk` shows strong correlation when k is low (consistent with Demirtas et al. (2020)'s observations of long mixing times), leveling off for higher k. This `flip_walk` correlation makes sense: the Markov-chain nature of `flip_walk` implies that samples with lag-k are at most k-flips apart from one-another. The explicit `random_triangulations_fair` algorithm discussed in Demirtas

[13]Every flip of a fine triangulation of a lattice polygon is a diagonal flip. That is, it replaces two simplices with two new ones.

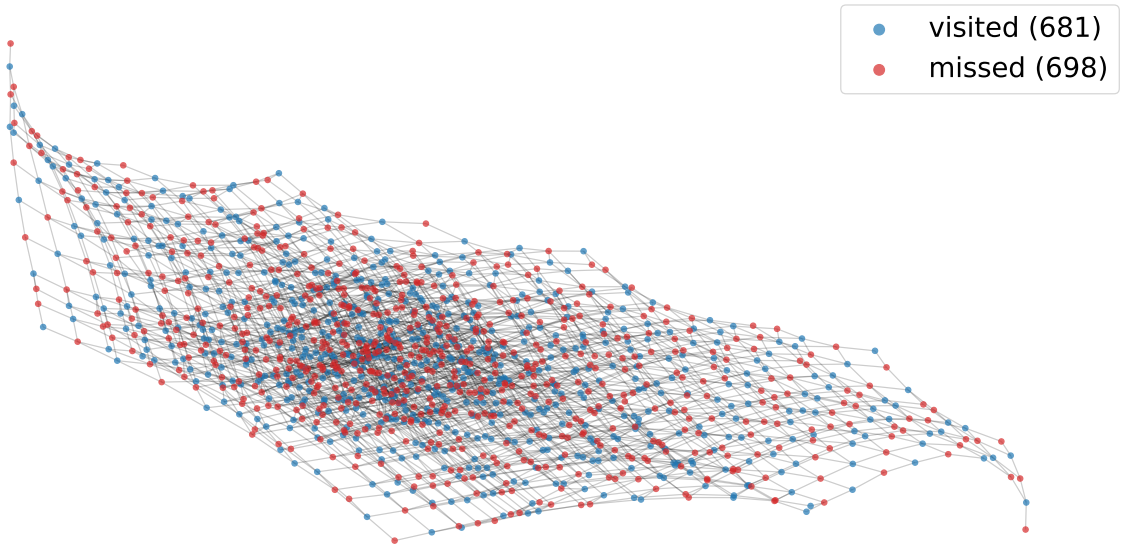

Figure 9: The flip graph of $\mathrm{conv}(\{(0,0),(2,3),(0,9)\})$, with nodes representing FRTs and edges representing flips. This graph is bipartite so a Markov-chain sampler such as `flip_walk` with an even number of steps between samples would only sample from one of two colors (red or blue). This example was found by happenstance on the first polygon tested.

et al. (2020) takes extra measures (modifications to heights) that may resolve this correlation, but this is anticipated to come at the expense of the already slow sampling rate, as we discuss below.

Sample rate splits the methods into four tiers (fig. 7): `pushing` is fastest at $\sim 40,000$ samples/second (since it does not require a regularity check); `grow2d`/`fast` follow at $500-650$ samples/second; dualGNN is slower (regularity + simplex-selection) at ∼200 samples/second[14]; `flip_walk` is slowest, at ~ 50 samples/second. All reported times include regularity checking (except for `pushing` which has guaranteed-regular outputs). The speed of `flip_walk` is set in part by the number of flips between samples; one could reduce this from 5 to a lower number, but that would worsen the correlation between samples. Here, we note that the CYTools variant of `random_triangulations_fair` is significantly slower than even `flip_walk`.

A second concern with `flip_walk` arises from the structure of the flip graph itself. Some flip graphs are bipartite (see fig. 9), meaning a walk with an even number of steps between samples will be unable to generate $\sim 50\%$ of all triangulations. This is resolvable by using an odd number of flips between samples. dualGNN does not face this problem: each inference call generates an independent sample.

Overall, `pushing` and `grow2d` are competitive samplers if bias is tolerated, especially `pushing`, which is very quick (but this method cannot, even in principle, generate some FRTs). `flip_walk` is a very effective untrained model, but it has non-trivial sample correlation and a sensitivity to walk-length choice. `fast` is generally not recommended — it is neither extremely fast nor very uniform. Finally, if uniformity is crucial, dualGNN outperforms all methods while having speed between `grow2d` and `flip_walk` (dualGNN has since become significantly quicker — see footnote 14).

[14] All timing numbers in this paper were measured with the experiment-era code (release `v0.0.1` in the repository). Subsequent releases include inference optimizations affecting only dualGNN: polygon sampling improved from ∼200 to ∼750 samples/second on the polygon of fig. 5 (RTX 5060 Ti), CPU inference sped up by $\gtrsim 4\times$, and end-to-end CY sampling (with the combination of FRTs into an FRST) now runs within $\sim 5\times$ of `fast`. The comparisons reported in the text are therefore conservative.

Sampler	# unique	# collisions	excess KL
`fast_c0.2`	$61,316$	8.12×10^9	6.44
`fast_c0.5`	$499,829$	3.49×10^9	5.26
`pushing`	$3,779,093$	1.24×10^8	1.82
`grow2d`	$7,321,880$	2.19×10^7	0.58
dualGNN	$9,895,669$	1.05×10^5	0.005
`flip_walk`	$9,924,852$	7.55×10^4	0.001
$1/N$	$9,932,320$	6.80×10^4	0

Table 1: Performance of FRT samplers on $[0,4]^2$, listed in order of increasing fairness. Notably, dualGNN here is the model trained on the polygon in fig. 5 applied zero-shot to $[0,4]^2$. The bottom row contains theoretical predictions for a true uniform sampler, for which there are $\#\text{unique} = N\big(1-(1-1/N)^M\big)$ and $\#\text{collisions} = M(M-1)/(2N)$. `flip_walk` is the most fair sampler, but dualGNN is not far behind despite being applied zero-shot.

2.2.2 Zero-shot Transfer

As we show here and in section 2.2.3, dualGNN generalizes across polygons. To demonstrate this, we begin by applying the model trained in section 2.2.1 zero-shot to the polygon $[0,4]^2$. This polygon has 25 lattice points and 4 vertices; each triangulation of it consists of 32 simplices. In all regards, then, sampling from $[0,4]^2$ is a harder task than that from the polygon in fig. 5 which has 19 lattice points and 3 vertices; each triangulation of the polygon in fig. 5 consists of 24 simplices. As aforementioned, these polygons also differ significantly in triangulation count: $[0,4]^2$ has $735,430,548$ FRTs (out of $736,983,568$ FTs) while the polygon in fig. 5 has only $405,706$ (out of $408,826$).

Generating the billions of samples needed for a complete uniformity measurement is outside our compute budget, so we instead draw 10^7 and assess uniformity within them. This is arguably a more representative test: for most applications in string theory, one wants a modest number of FRTs sampled from 2-faces with astronomical counts of FRTs — the large number of CYs come from different ways of grouping these samples together Berglund et al. (2025); MacFadden et al. (2025). The small sample pool compared to the total number of FRTs means that even a truly uniform (but finitely drawn) sampler shows significant KL divergence compared to the flat distribution, making KL divergence a less informative diagnostic in this case (see fig. 10). Still, `flip_walk` and dualGNN show the lowest KL divergence, only < 0.005 above the noise floor (see table 1; set by our finite number of samples). `flip_walk` has the lowest KL.

To better discriminate bias, we also study the total number of collisions (different samples giving the same triangulation). For M uniform samples out of a pool of N items, any pair $\binom{M}{2}$ is expected to collide at probability $1/N$. Thus, we expect a total number of collisions

$$\#\text{collisions} = \binom{M}{2}\frac{1}{N} = \frac{M(M-1)}{2N}. \tag{2.5}$$

For our case, with $M = 10^7$ and $N = 735,430,548$, we thus predict $67,987$ collisions (out of $\sim 5 \times 10^{13}$ possible ones). This diagnostic shows bias in all samplers (see table 1), with `flip_walk` and the zero-shot dualGNN having the closest number of collisions to the uniform prediction. Again, `flip_walk` shows the lowest bias here, but the zero shot dualGNN is not far behind.

This is very strong performance from dualGNN given that this model had only been trained for $O(\text{hours})$ on a *different* polygon with significantly different shape, point count, and triangulation count (i.e., that from fig. 5). This suggests that the architecture generalizes strongly across polygons.

2.2.3 Multiple Polygons

The ultimate goal of dualGNN is to be a general-purpose FRT sampler. We describe such a model in this section. Trained across 271 polygons and fine-tuned with REINFORCEWilliams (1992), it achieves the most

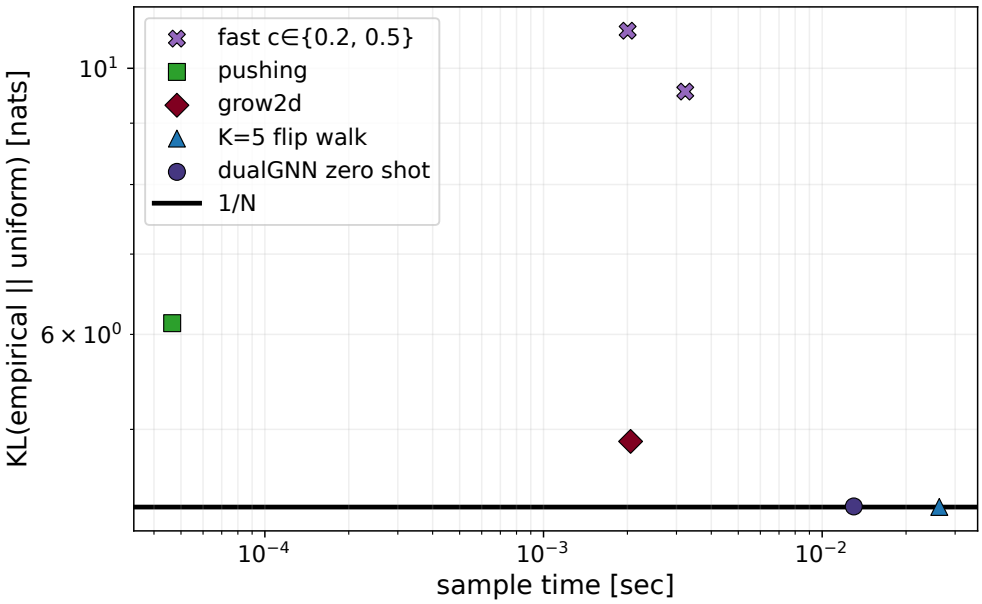

Figure 10: The dualGNN sampler, trained on the polygon in fig. 5, applied to $[0, 4]^2$. This polygon has $735,430,548$ FRTs, many more than we could sample in our computational budget (10^7). Both `flip_walk` and dualGNN achieve near uniform pools of samples, measured against the noise floor due to the finite number of samples. `grow2d` and `pushing` show moderate bias but significantly increased sampling rates. Finally, `fast` shows the largest bias despite being significantly slower than `pushing`.

uniform sampling of any method tested across $N_{\text{pts}} \leq 18$ and is consistent with uniformity at $N_{\text{pts}} = 40$ within the resolution granted by our sample budgets.

We maintain the same hyperparameters as before (32-dimensional feature vectors, 16 message-passing rounds). What we change is the training: instead of training on (sometimes bootstrapped) pools of fine triangulations of a single polygon, we train on bootstrapped pools from *randomly chosen* polygons. Explicitly, we generate 116 randomly chosen polygons to start with $12 \leq N_{\text{pts}} \leq 40$ and build initial pools by either fully enumerating the triangulations (for $N_{\text{pts}} \leq 17$) or sampling an initial pool of $10,000$ FTs using `grow2d`. To better enable multi-polygon learning, we also modify the exploration rounds: in each such round, with probability 80%, we pick an existing polygon from the training pool; otherwise (20% chance) we generate a new random polygon with $5 \leq N_{\text{pts}} \leq 40$. In our ~ 5.5 hour training (on a single 5060 Ti) over $500,000$ steps, this led to 271 polygons in total. Each new polygon is seeded with $2,000$ `grow2d` samples.

To evaluate this trained model, we apply it to 20 polygons from outside the training pool (see fig. 11) with $11 \leq N_{\text{pts}} \leq 18$, generating $200,000$ samples from each polygon. The supervised learning leads to moderate uniformity (orange pluses in fig. 12), but it is not up to our standards. This is likely because the cross-entropy objective is only an indirect proxy for what we actually want (a uniform sampler over complete triangulations). Cross-entropy matches per-step conditional probabilities to estimated targets, but biases in the training pool *do* affect the uniformity of the trained model. To directly optimize the triangulation distribution, we fine-tune dualGNN with REINFORCE Williams (1992) using an entropy-maximizing reward. Explicitly, we perform $10,000$ steps on the training polygons, each step a batch of four full rollouts. After the rollouts, the probability of each triangulation being sampled is computed as the product of each simplex's conditional probability. If the triangulation is regular, we then assign reward $-\log \text{prob}(T)$, otherwise the triangulation receives reward -2 (penalty magnitude chosen arbitrarily since no regular triangulation gets a negative reward). This maximizes entropy, rewarding the policy for generating regular triangulations it deems unlikely, driving the distribution toward uniformity. We clip gradients by magnitude 1 and use a learning rate of 3×10^{-5}. This RL post-training takes ~ 2 additional hours (~ 7.5 hours total training) and leads to significantly more uniform samples (green circles in fig. 12). Comparing to other reference samplers

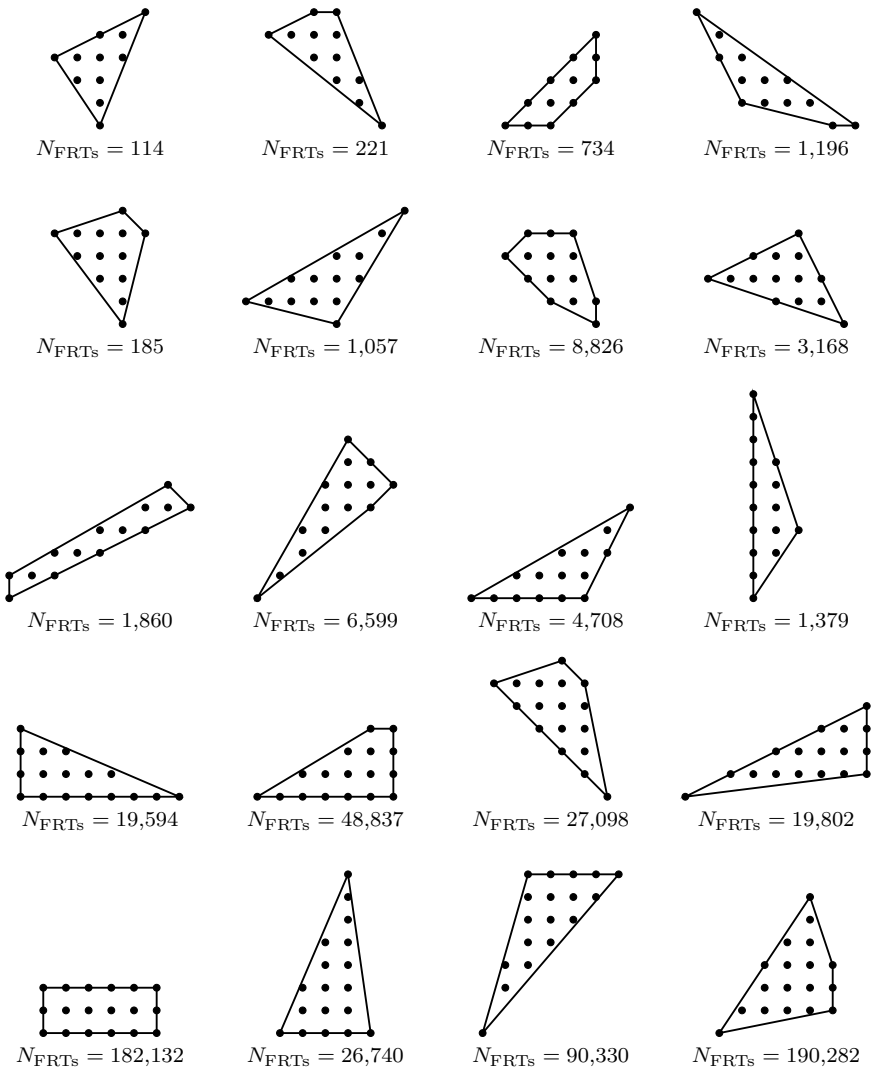

Figure 11: The 20 out-of-distribution polygons used in fig. 12, with $11 \leq N_{\text{pts}} \leq 18$, sorted by N_{pts}.

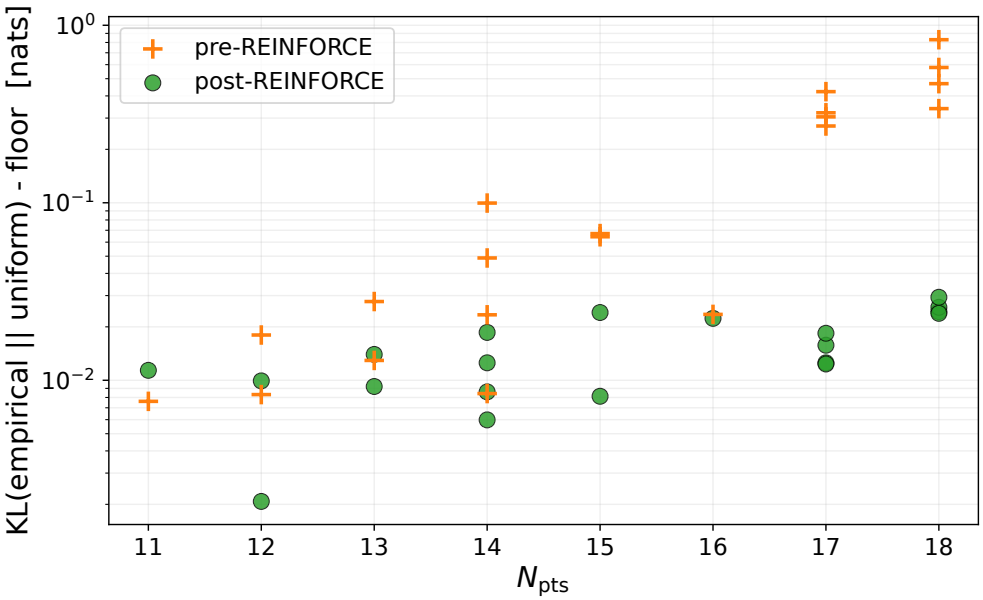

Figure 12: Pre- and post-REINFORCE finetuning on the dualGNN autoregressive sampler that was trained on multiple polygons. All of these markers correspond to polygons not seen in the training (see fig. 11), $200,000$ samples each. In all cases except the smallest polygon, REINFORCE caused more uniform samples, oftentimes significantly so.

post this fine-tuning, we find that dualGNN is the most uniform studied (see fig. 13; tied with `flip_walk` at $N_{\text{pts}} = 18$) and, in contrast to `flip_walk`, shows no autocorrelation (fig. 14).

One subtlety deserves comment: our reward is defined over rollouts (orderings of simplex placements) while uniformity is desired over triangulations, and a single triangulation is reachable by many rollouts. GFlowNets Bengio et al. (2021) were designed for precisely this many-paths setting. In our 2D setting, however, this machinery is not strictly necessary: every fine triangulation of a fixed polygon contains the same number n of simplices and every ordering of a triangulation's simplices is a feasible rollout, so each triangulation corresponds to exactly $n!$ rollouts. Uniformity over rollouts therefore implies uniformity over triangulations, and our entropy-reward REINFORCE targets the desired distribution directly. Note, however, that our architecture treats a triangulation's simplices as a sequence (in the selection order) rather than a set, and so does not build in their order-invariance; a GFlowNet would respect this symmetry structurally. This is a cost in inductive bias rather than in the uniformity itself. Additionally, GFlowNet objectives become attractive for non-uniform target distributions, higher dimensions, or learning non-uniform distributions (the purpose of Berglund et al. (2025); MacFadden et al. (2025); a goal listed in Yip et al. (2025)); we leave a direct comparison to future work.

We also apply this same model to larger polygons, each with $N_{\text{pts}} = 40$ (see fig. 15). The large triangulation counts for such large polygons (with upper bounds ranging from 3.7×10^{19} to 5.9×10^{20} FTsAnclin (2003)) make it significantly more difficult to measure bias. Even significantly biased samplers can show low collision counts, meaning that they would have KL divergences nearing the noise floor (due to our finite number of samples). We address this, in part, by again using the collision count itself as a diagnostic since it was better suited to sparse samples in section 2.2.2.

Across the four $N_{\text{pts}} = 40$ polygons, dualGNN is the only sampler to have no collisions across the $100,000$ draws (see table 2). This is consistent with the above rough estimates on the number of FRTs for these polygons, $\gg 10^9$. This lack of collisions is a necessary, not sufficient, test for dualGNN's uniformity. No other method passes this test: `grow2d` is the only other sampler that achieves 0 collisions for one polygon, but it has nonzero collisions for all other polygons. All other methods collide on all polygons. By inverting the predicted number of uniques for a uniform sampler, $\#\text{unique} = N\big(1 - (1 - 1/N)^M\big)$, we can get a rough

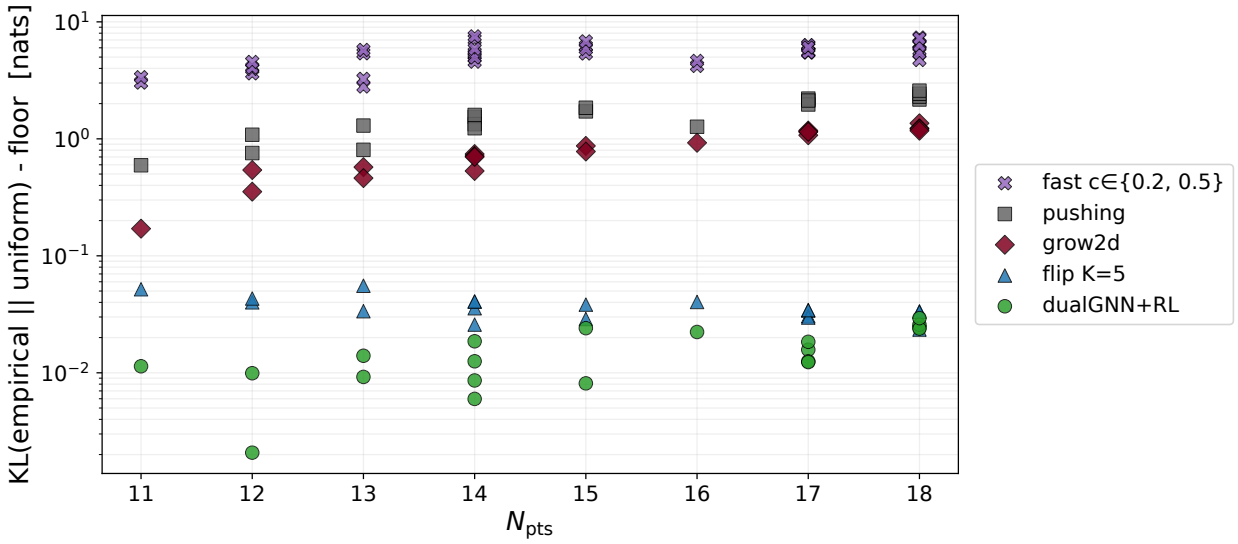

Figure 13: Comparison of the uniformity of samples for 200,000 samples on the polygons in fig. 11. In all cases, dualGNN is the most uniform, except the largest $N_{\text{pts}} = 18$ at which it is equal to `flip_walk` despite being $\sim 4\times$ faster than `flip_walk`.

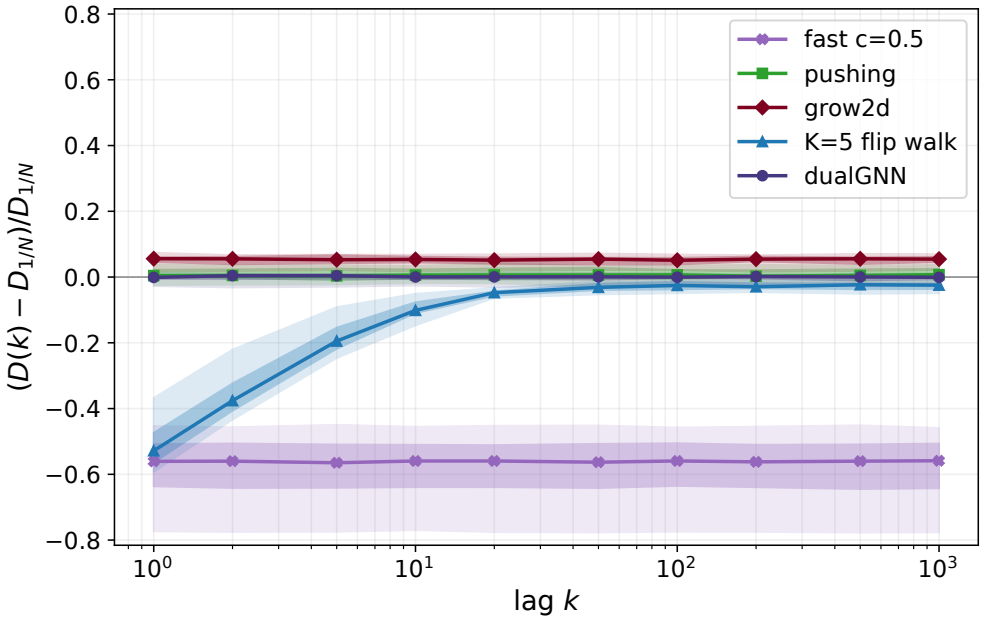

Figure 14: Autocorrelation of the various samplers over 200,000 samples on the polygons in fig. 11. As with fig. 8, we use the symmetric difference between the triangulations as a lower bound on the flip distance, with a curve for the mean value and bands indicating the spread. dualGNN and `pushing` are both consistent with a uniform distribution ($y = 0$) for all polys while `flip_walk` shows significant correlation for low lags. `grow2d` and `fast` both show constant flip distances, which would be consistent with uniform, but they average to a different distance than what a uniform sampler would predict.

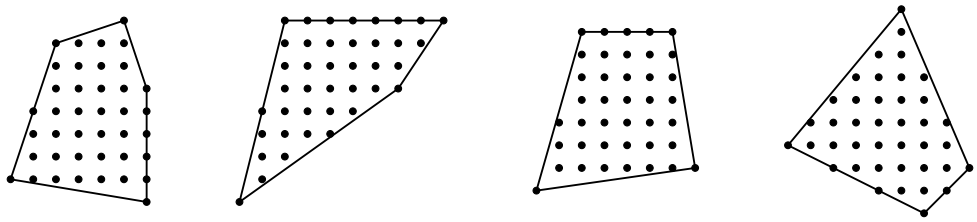

Figure 15: Four out-of-distributions polygons at $N_{\text{pts}} = 40$ used for post-REINFORCE inference.

estimate of an 'effective' population that each sampler is sampling out of (if it were uniform). For the biased samplers (`fast`), this is as low as 10^5; for other samplers, this is consistently $\lesssim 10^9$. We stress: the lack of collisions is not a proof that dualGNN is uniform, just the passing of a necessary test that no other samplers passed, not even often-competitive `flip_walk`.

	Polygon P_1			Polygon P_2			Polygon P_3			Polygon P_4		
Sampler	# unique	# coll.	N_{eff}	# unique	# coll.	N_{eff}	# unique	# coll.	N_{eff}	# unique	# coll.	N_{eff}
`fast_c0.2`	94,702	5,298	9.1×10^5	81,164	18,836	2.3×10^5	87,123	12,877	3.5×10^5	75,995	24,005	1.7×10^5
`fast_c0.5`	98,373	1,627	3.0×10^6	96,584	3,416	1.4×10^6	99,019	981	5.1×10^6	94,968	5,032	9.6×10^5
`pushing`	99,993	7	7.1×10^8	99,989	11	4.5×10^8	99,991	9	5.6×10^8	99,988	12	4.2×10^8
`grow2d`	100,000	0	∞	99,997	3	1.7×10^9	99,996	4	1.3×10^9	99,998	2	2.5×10^9
`flip_walk`	99,998	2	2.5×10^9	99,995	5	1.0×10^9	99,997	3	1.7×10^9	99,997	3	1.7×10^9
dualGNN	100,000	0	∞	100,000	0	∞	100,000	0	∞	100,000	0	∞

Table 2: Performance of FRT samplers on the $N_{\text{pts}} = 40$ polygons. Each polygon block reports the number of unique triangulations recovered (# unique), number of collisions (# coll.), and the effective population N_{eff} obtained by inverting $\#\text{unique} = N\big(1-(1-1/N)^M\big)$ for N at $M = 100{,}000$. Note: for 0 collisions, the inversion gives $N_{\text{eff}} = \infty$, which is only an artifact of our small sample count. Recall that there are strict upper bounds ranging from 3.7×10^{19} to 5.9×10^{20} FTs (hence FRTs) for these polygons.

We also return to the autocorrelation of the samplers. This is arguably more important for these large polygons: as polygons get more FTs, their flip graphs become correspondingly larger and there is increased risk for a sampler that makes local explorations (i.e., `flip_walk`) to only explore a small region of triangulations. As can be seen in fig. 16, only dualGNN is consistent with a uniform sampler. `flip_walk` again shows significant correlation for low lags while the other methods show constant distances, but inconsistent with dualGNN and the large-k limit of `flip_walk` (which are consistently the most uniform samplers tested so far).

Altogether, the multi-polygon dualGNN model is the only sampler consistent with uniform sampling across all our diagnostics. In all tests above, dualGNN has consistently lower KL divergences than the other methods (tying with `flip_walk` in some cases), consistent collision rates, and consistent flip distances with a uniform sampler. We stress: these results are all for polygons unseen in the training. dualGNN's uniformity on unseen polygons suggests that it learns the geometry of the problem rather than memorizing a training distribution.

3 Application: CY Sampling

First we recall from section 1 the connection to Calabi-Yau threefolds (CYs). This paper was motivated by the application of generating CYs using Batyrev's construction Batyrev (1994). This construction maps a fine, regular, and 'star' triangulation (FRST) of a 4D reflexive polytope (all such polytopes enumerated in the Kreuzer-Skarke database Kreuzer & Skarke (2000), directly accessible in CYTools) to a CY. As argued in the introduction, this map is many-to-one due to any two FRSTs with the same 2-face restrictions mapping to homotopy-equivalent CYs. This motivated our prior development of the 'NTFE algorithm' in MacFadden

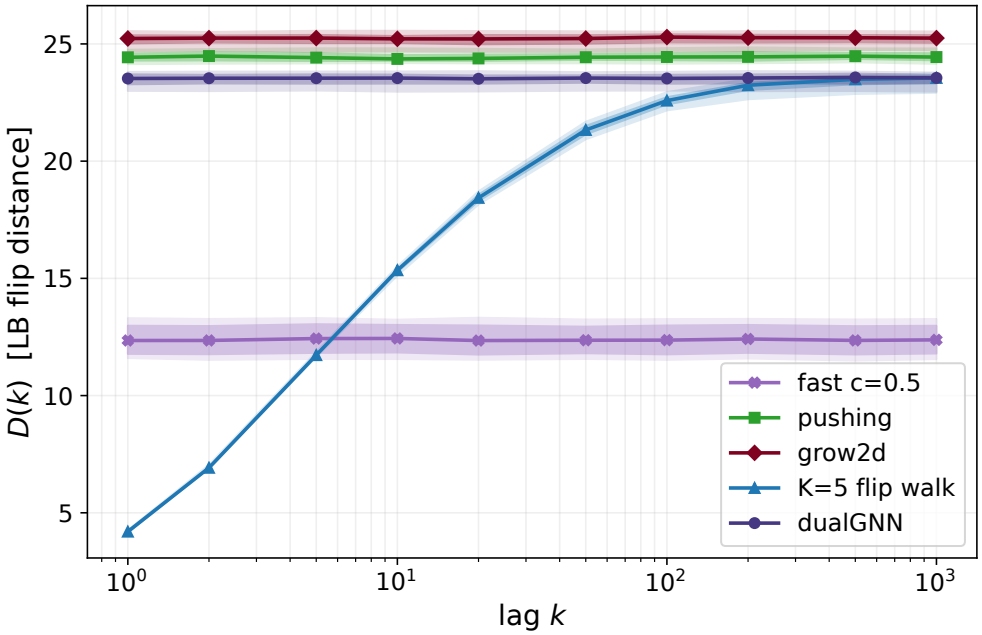

Figure 16: Autocorrelation of the various samplers over 100,000 samples on the polygons in fig. 15. As with previous plots, we use the symmetric difference between the triangulations as a lower bound on the flip distance. Since we do not have counts of FRTs for these polygons, we cannot compute $D_{1/N}$ so we just plot the raw lower bounds $D(k)$ on the y-axis. This means that we cannot definitively say that dualGNN is consistent with uniform sampling (since we do not have $D_{1/N}$) but we can (a) identify that `flip_walk` still shows the correlation for small lags, (b) see that the flip distances across methods show the same relative pattern as in figs. 8 and 14, and (c) see that the large-k limit of `flip_walk` (presumed uniform) matches dualGNN. This is consistent with expectations if dualGNN were uniform.

(2023) which constructs FRSTs via their 2-face triangulations, thus sidestepping the redundancy. This construction and the NTFE algorithm are both implemented in CYToolsDemirtas et al. (2022).

Since uniformity in the CY samples is a key goal, we review the uniformity of rejection sampling à la MacFadden (2023) here. For a given polytope Δ with 2-faces $f_1, f_2, \ldots, f_{N_{2D}}$, the NTFE algorithm MacFadden (2023) accepts triangulations $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{N_{2D}}$ and outputs an FRST with said 2-face restrictions if one exists. If an FRST exists, one says $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{N_{2D}}$ 'extends' MacFadden (2023). This FRST can be interpreted as a representative of a 2-face equivalence class (FRSTs mod identical 2-face restrictions) which MacFadden (2023) originally called an 'NTFE'; it will be convenient to also follow MacFadden et al. (2025) and call $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{N_{2D}}$ the 'DNA' of this NTFE. We want to show that

$$P(\mathcal{T}_i) = \frac{1}{\mathcal{N}_i} \implies P(\text{NTFE}_j|\text{DNA extends}) = \frac{1}{\#\text{NTFEs}}, \tag{3.1}$$

using notation where $\mathcal{N}_i$ is the number of FRTs for the 2-face f_i. Observe that $P(\text{NTFE}_j \cap \text{extends}) = P(\text{DNA}_j)$ for DNA_j the DNA associated to NTFE_j since the algorithm is a bijection between extendable DNA and NTFEs. Thus $P(\text{NTFE}_j \cap \text{extends}) = \prod_{1 \leq i \leq N_{2D}} P(\mathcal{T}_i)$ for $\mathcal{T}_i$ the triangulation of f_i corresponding to NTFE_j. This, under our assumption on $P(\mathcal{T}_i)$, immediately gives us our result

$$P(\text{NTFE}_j|\text{extends}) = \frac{1}{P(\text{extends})} \prod_{1 \leq i \leq N_{2D}} \frac{1}{\mathcal{N}_i} = \frac{1}{P(\text{extends}) \cdot \#\text{DNAs}} = \frac{1}{\#\text{NTFEs}}. \tag{3.2}$$

This is why MacFadden (2023) is a reduction: by uniformly sampling 2-face triangulations, it directly gives you uniform samples over NTFEs/CYs.

3.1 Comparison to CYTransformer

We begin by briefly comparing to CYTransformer Yip et al. (2025)[15]. Since CYTransformer generates 4-simplices directly, its output space includes the exponentially redundantDemirtas et al. (2020) collection of FRSTs sharing 2-face restrictions. dualGNN, in contrast, sidesteps this redundancy by generating DNA directly. This leads to dualGNN showing strong performance in Yip et al. (2025)'s diagnostic of the average number of NTFEs generated at different $h^{1,1}$. We emphasize that this comparison primarily measures the value of the 2-face reduction rather than relative model quality: any 4D sampler must contend with the exponential FRST redundancy that dualGNN sidesteps. We demonstrate dualGNN's performance on the 200 polytopes that Yip et al. (2025) studied ($5 \leq h^{1,1} \leq 10$) and on 200 more for each of $h^{1,1} = 12, 14, 16$ (first 200 at each $h^{1,1}$ in Kreuzer-Skarke order). These Hodge numbers are relatively small, so we can enumerate all NTFEs using MacFadden (2023) to give a true 1/#NTFE uniform sampler comparison.

In all cases for which Yip et al. (2025) presented data ($h^{1,1} \leq 10$), dualGNN generates more NTFEs with fewer samples (NTFE curves from Yip et al. (2025) come from their digitized figure). See fig. 17. In fact, across all Hodge numbers, dualGNN is consistent with a uniform sampler (within noise) while CYTransformer undersamples NTFEs relative to the uniform reference (particularly at $h^{1,1} = 8, 9$, and 10). This makes sense: because CYTransformer samples in the redundant FRST space rather than via a 2-face encoding, the fraction of triangulations that are NTFE-distinct falls as $h^{1,1}$ grows, since FRSTs are exponentially-with-$h^{1,1}$ redundant compared to NTFEs. This is precisely the regime ($h^{1,1} \gg 10$) where such samplers are most needed, and it is exactly what the 2-face reduction of MacFadden (2023) is designed to address.

3.2 Sampling at High-$h^{1,1}$

We also push the CY sampling to higher Hodge numbers for which MacFadden (2023) cannot exhaustively enumerate all NTFEs. First, as a proof of concept, we study the following $h^{1,1} = 86$ polytope

$$\Delta_{86} = \text{conv}\begin{bmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 3 & -1 & -1 & -1 & 3 \\ -1 & -1 & 3 & -1 & -1 & -1 & 3 & -1 \\ -1 & 3 & -1 & -1 & -1 & 3 & -1 & -1 \end{bmatrix}. \tag{3.3}$$

We choose Δ_{86} because each of its 2-faces contains exactly $N_{\text{pts}} = 15$, so we can validate dualGNN's uniformity against full enumeration. While this is a non-negligible $h^{1,1}$, significantly higher than what previous works could sample (especially because the number of triangulations depends exponentially on $h^{1,1}$), this is not a very demanding application of dualGNN. One can fully enumerate the FRTs of these 2-faces in short order: there are two distinct geometries, $\text{conv}(\{(0,0),(0,4),(4,0)\})$ and $\text{conv}(\{(0,0),(0,4),(2,0),(2,4)\})$ with $7,422$ and $12,170$ FRTs respectively (see fig. 18). As a more demanding test, we also study the following $h^{1,1} = 128$ polytope

$$\Delta_{128} = \text{conv}\begin{bmatrix} 1 & -11 & -11 & 1 & 1 & 1 & 13 \\ 0 & -4 & -4 & 0 & 0 & 2 & 6 \\ 0 & -6 & -6 & 2 & 2 & 0 & 8 \\ 0 & -12 & -6 & 0 & 6 & 0 & 12 \end{bmatrix} \tag{3.4}$$

for which each 2-face has $N_{\text{pts}} \leq 35$ (see fig. 19). In principle there is nothing stopping us from studying arbitrarily large $h^{1,1}$; we only stop at $h^{1,1} = 128$ since our uniformity diagnostics already begin to struggle at $N_{\text{pts}} \approx 40$, so it would be significantly harder to validate the uniformity of larger polytopes.

We begin with Δ_{86}, which only has two distinct 2-face geometries (fig. 18). We generate 10^6 FRTs of each 2-face with the dualGNN model from section 2.2.3. For 2-faces with $N_{\text{pts}} \leq 17$, we find a maximum KL divergence (compared to a flat distribution) of 0.016 across such 2-faces; a uniform, but still 10^6-draw sampler achieves a maximum KL divergence of 0.006. This suggests that dualGNN is indeed nearly uniformly sampling FRTs of these 2-faces, as should be expected from section 2.2.3.

[15] We use slightly different diagnostics: Yip et al. (2025) assess sampling quality via height-space representativeness while we hold dualGNN to the stronger criterion of per-triangulation uniformity, which implies representativeness but not conversely.

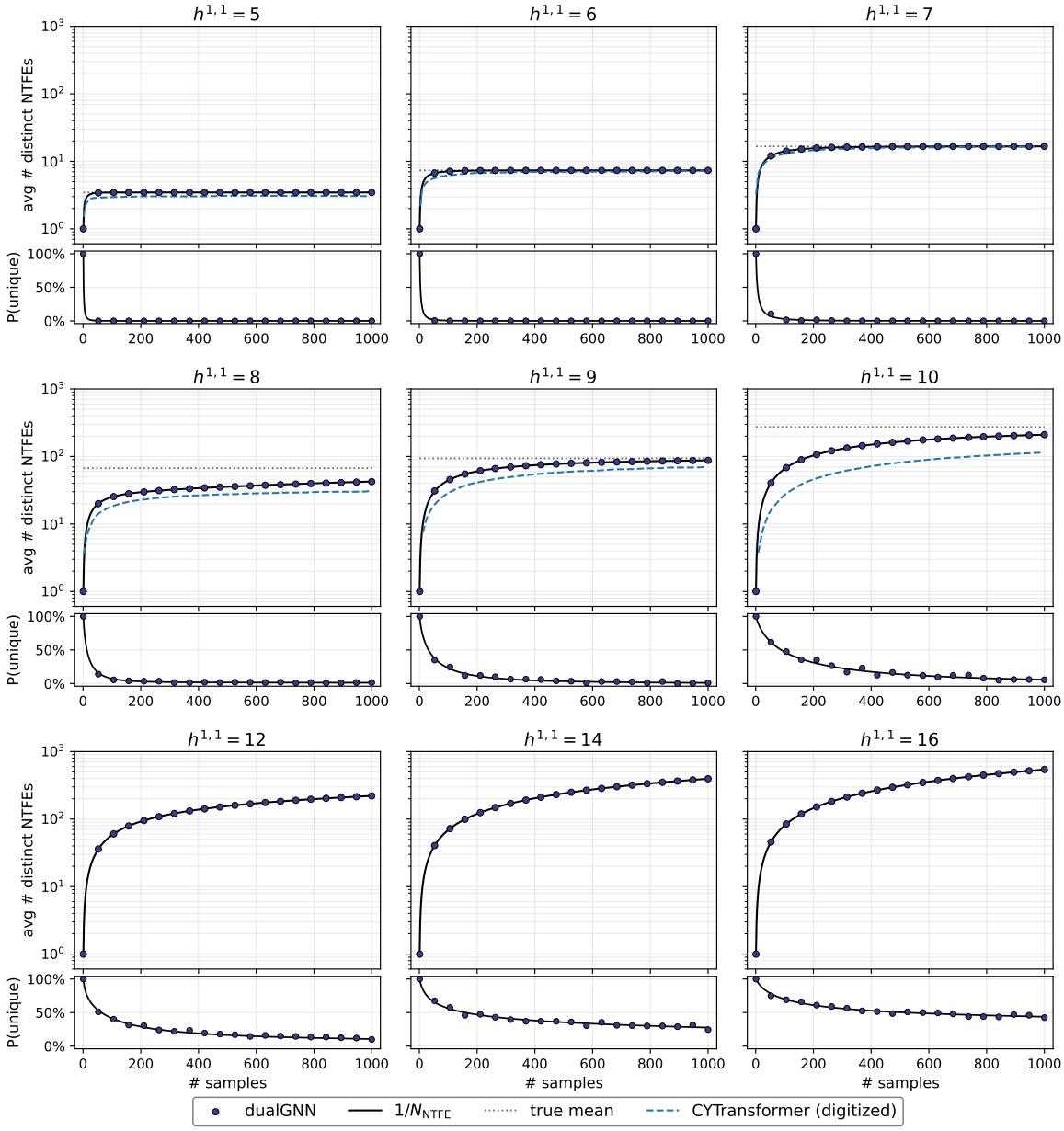

Figure 17: Recreation of Yip et al. (2025)'s figure 4, extended to larger Hodge numbers $h^{1,1} = 12, 14, 16$. At all Hodge numbers, dualGNN generates NTFEs at a rate (purple dots) consistent with a true $1/N_{\text{NTFE}}$ sampler (black line). For Hodge numbers $h^{1,1} \leq 10$, we compare directly to the polytopes studied in Yip et al. (2025) (obtained via private communication), consistently obtaining more NTFEs at every sample count. This advantage is primarily attributed to the 2-face encoding rather than the relative quality of the underlying models. The CYTransformer curves come from their digitized figure 4.

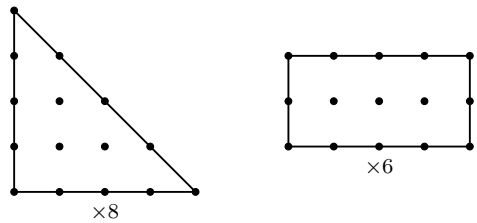

Figure 18: The two distinct 2-faces of the $h^{1,1} = 86$ polytope, labeled with their multiplicity in the polytope. The triangle (left) has $7,422$ FRTs; the rectangle (right) has $12,170$.

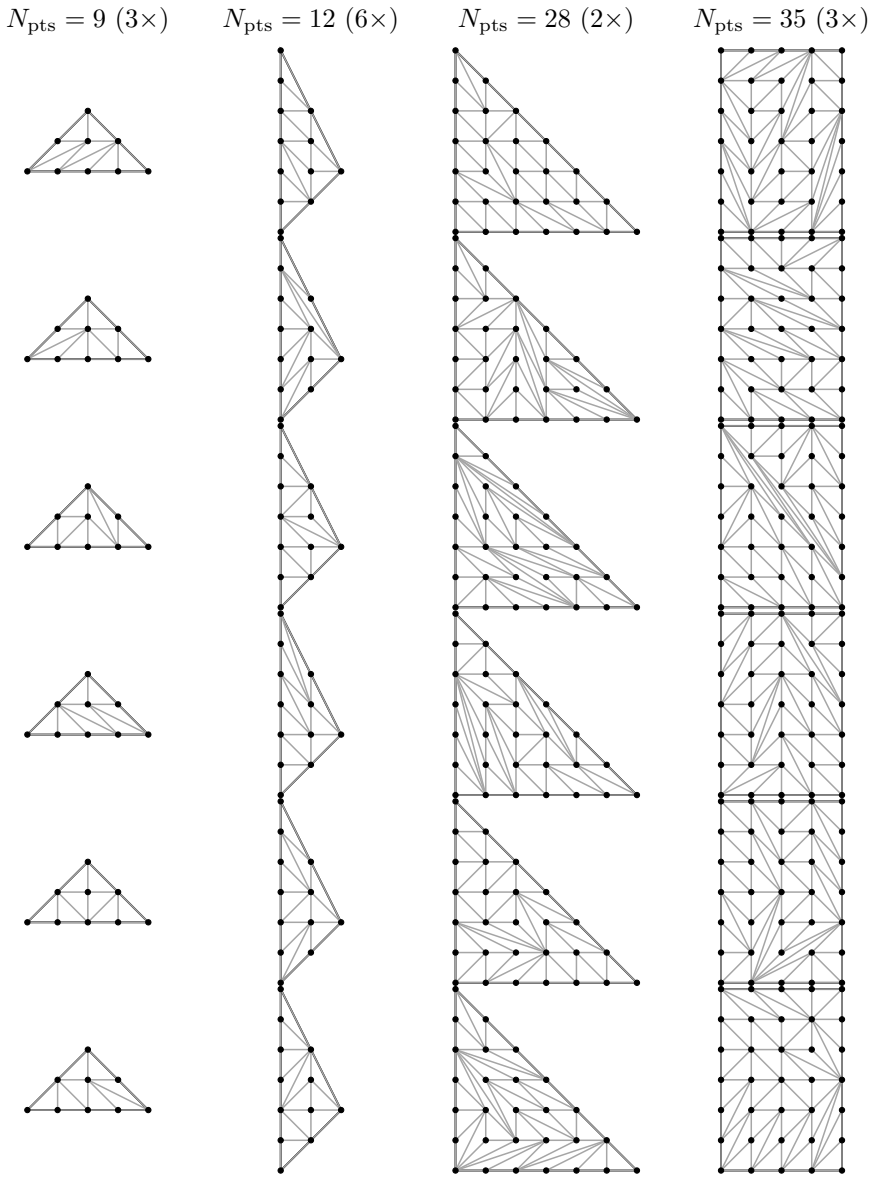

Figure 19: Example triangulations of the four unique 2-faces of Δ_{128}. These triangulations are all generated by the trained dualGNN sampler.

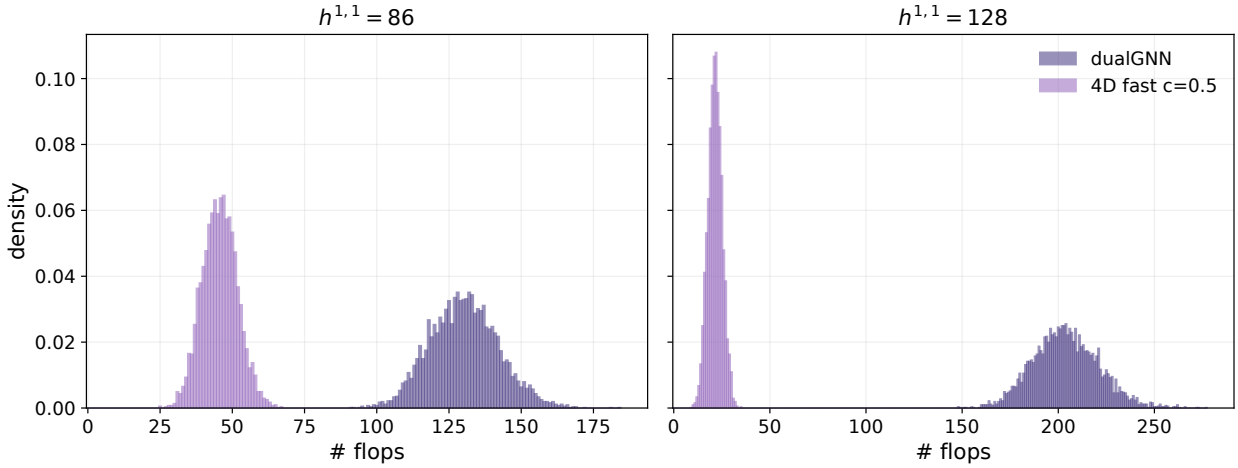

Figure 20: Diversity of sampled CYs, measured by pairwise flop distance (upper bound via a linear path through height-space). In dark purple, we plot the samples from dualGNN, combined via MacFadden (2023). In light purple, we plot the samples from `fast` applied to the 4D polytope. Left: for Δ_{86}, `fast` generates CYs that are between 25 and 67 flops from one-another (mean 45.7). In contrast, the dualGNN alternative generates samples that are between 91 and 184 flops from one-another (mean 130.2). Right: Δ_{128}, `fast` generates CYs that are between 9 and 35 flops from one-another (mean 21.5) while dualGNN generates samples between 148 and 277 flops from one-another (mean 204.0). Larger flop distances indicate less concentrated samples; this comparison probes diversity, not uniformity.

From the dualGNN samples, we apply MacFadden (2023) to generate $10,000$ FRSTs of Δ_{86}, all of which happen to have distinct 2-face restrictions. By our arguments in section 3 and by the demonstrated uniformity of the 2-face triangulations, this is a uniform sample of CYs mod 2-face equivalences (out of the pool of $< 2.99 \times 10^{55}$). No prior work has uniformly sampled such CYs at Hodge numbers close to $h^{1,1} = 86$ before. These CYs are potentially inequivalent, not provably inequivalent; certifying full homotopy-inequivalence is a strictly harder problemGendler et al. (2023); Chandra et al. (2024), only done up to $h^{1,1} = 5$ with partial results at $h^{1,1} = 6$.

With uniformity addressed above, we turn to a physically motivated measure of sample diversity: flop distance. A 'flop' is a local geometric transition between two Calabi-Yau threefolds; the flop distance between two CYs is the minimum number of such transitions needed to transform one into the other, providing a discrete metric on the space of CYs. Explicitly, we measure an upper bound on the number of flops between these CYs, obtained via taking a linear trajectory in height-space using `regfans` MacFadden (2026); MacFadden & Sheridan (2025) and counting the flop transitions. We compare to a sample of $10,000$ FRSTs from `fast` in the left of fig. 20: from the dualGNN samples, we observe a mean flop count of 130.2, significantly higher than the mean 45.7 of the `fast` samples. That is, the dualGNN-sampled CYs are much more diverse than the de facto standard of `fast`-sampled CYs.

The more interesting example is that of Δ_{128}, for which one cannot fully enumerate the 2-face FRTs (2-faces have up to $N_{\text{pts}} = 35$; see fig. 19). Here, we perform analogous tests: we generate 10^4 FRTs of each 2-face and validate, for each 2-face with $N_{\text{pts}} \leq 17$, that the KL divergence is low (max is 0.012; uniform sampler with the same sample count has max KL divergence of 0.008). For the larger 2-faces, we confirmed that the dualGNN samples had no collisions, consistent with uniformity (but not sufficient to show it). With the uniformity of each 2-face argued, we likewise construct $10,000$ FRSTs of Δ_{128} by MacFadden (2023). Again comparing to `fast`, we see (right side of fig. 20) that the dualGNN samples have a mean flop count of 204.0 while the biased `fast` reference has only 21.5.

`fast` is significantly faster than the dualGNN construction (> 7 CYs/sec vs ~ 0.1 CYs/sec, but we remind that dualGNN has since become significantly quicker — see footnote 14) but (a) typically the CY generation is not rate limiting due to the high cost of most downstream applications and (b) good data (uniform samples) is

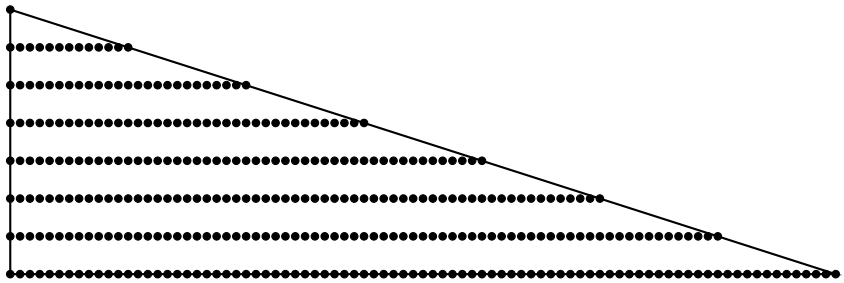

Figure 21: The polygon $\mathrm{conv}(\{(0,0),(84,0),(0,7)\})$, the largest polygon occurring in our string theory applications (as a 2-face of the only $h^{1,1} = 491$ 4D reflexive polytope). This polygon has between 3.90×10^{167} and 1.96×10^{180} FRTs.

typically much more valuable than biased data. We expose this sampler as `random_triangulations_gnn` in CYTools, so one can easily choose between these samplers. To our knowledge, this is the first demonstration of uniform CY sampling at Hodge numbers at $h^{1,1} = 86$, significantly above the previous $h^{1,1} = 10$. We hesitate to also claim this for $h^{1,1} = 128$, despite showing near uniform diagnostics, because our sampling resolution is worse here.

4 Limitations

dualGNN's architecture scales to polygons of any practical size — even the largest polygon of interest (the largest 2-face of $h^{1,1} = 491$, fig. 21) generates a graph of only 69,416 nodes. One can even run the code in the associated repo on this polygon, though it is far out of distribution and the $K = 16$ message-passing rounds are likely too few. The primary limit is in *validating* uniformity at scale: this polygon has $N_{\text{pts}} = 344$ while our diagnostics start to lose resolution around $N_{\text{pts}} \approx 40$.

Additionally, dualGNN's strongest architectural guarantee, that every forward pass produces a valid and fine triangulation, is specific to 2D. The construction relies on the fact that any uncovered region of a partial 2D triangulation always admits a fine completion using existing lattice points; this is not true in higher dimensions. dualGNN can still operate in higher dimensions: circuits generalize naturally as the underlying combinatorial object, and the symmetry invariance follows. The specific $\mathbf{C}_{ab}$ encoding would need to be rebuilt (the 4D vector with 'left'/'right' ordering is 2D-specific), but that only requires engineering. The every-rollout-produces-an-FT guarantee is simply lost — it relies on a geometric fact specific to 2D. This matters for applications like vex triangulations MacFadden & Sheridan (2025) which currently require operating on 4D geometries directly (no algorithm like MacFadden (2023) exists yet for these geometries).

Relatedly, the circuit encoding is sufficient to expose regularity, as demonstrated by the classifier of section 2.1, but the autoregressive sampler does not perfectly enforce it. The strategy to best ensure regular triangulations is unclear: improve training or architecture? Section 2.1 had a nearly identical (strictly simpler) architecture and could easily extract regularity. That said, the architecture could likely also be improved by adding global attention à la Graphormer Ying et al. (2021). Evidence for this is that our transformer experiments in appendix C used a similar training regime to our dualGNN autoregressive sampler but were better at targeting regularity (see fig. 27 for a good irregular sampler).

Finally, dualGNN is empirically uniform, with no theoretical guarantees. There is a provably uniform algorithm for sampling FTs (and hence FRTs via rejection sampling): Kaibel, V. and Ziegler, G.M. (2003). This algorithm is not generally applicable due to its exponential scaling on the polygon width. For example, in MacFadden et al. (2026), we applied the counting variant of this algorithm to the polygon in fig. 21, but that took ~ 57 core days (serially). While dualGNN currently is not applicable for this polygon ($K = 16$ message passing rounds are too few), its time scales linearly with K and one must set K at least as large as the maximum diameter of a dual graph of the polygon. This is significantly milder scaling than exponentially with the width. In practice, using the existing dualGNN code in the associated repo (albeit with the

optimistic $K = 16$), one can sample a triangulation in $\sim 60s$ for graph creation (amortized) and $\sim 75s$ for rollout (although the rollout time is expected to be an underestimate) on this $h^{1,1} = 491$ polytope.

5 Conclusion

We introduced dualGNN, an autoregressive GNN which encodes all fine triangulations of a lattice polytope Δ via a generalization of the dual graph of a triangulation. By encoding certain 'dependency vectors' λ corresponding to 'circuits', one both better represents the combinatorics of the triangulation (the oriented matroid) and the regularity of the associated triangulations. We showed that such a GNN can extract the regularity signal in its edges and can operate as an autoregressive sampler of fine, regular triangulations consistent with uniformity. This autoregressive sampler generalizes zero-shot to other polytopes due to its symmetry invariance and N_{pts}-independent structure, making a single ~ 92k parameter model generally applicable to unseen polygons. The small size of the model enables very quick training on consumer hardware, O(hours). Overall, dualGNN was the only sampler consistent with uniformity across all diagnostics, with an architecture that scales to any polygon size of practical interest.

We applied this model to the task of generating CYs at Hodge numbers $h^{1,1} = 86$ and $h^{1,1} = 128$. For $h^{1,1} = 86$ our diagnostics are strong enough to validate uniformity (but this case does not require dualGNN since one can fully enumerate its FRTs for the 2-faces); for $h^{1,1} = 128$ dualGNN passes our uniformity diagnostics (within their resolution limits), but we stress that our diagnostics are weaker here. This reach is enabled both by the 2-face reduction of MacFadden (2023) and by dualGNN's polygon-general uniform sampling.

More broadly, this model demonstrates a GNN-compatible encoding for realizable oriented matroids (those representable as point or vector configurations Björner et al. (1999); De Loera et al. (2010)). Oriented matroids are common as the underlying structure of various disparate mathematical domainsBjörner et al. (1999), including directed graphs, linear programming, hyperplane arrangements, convex polytopes, and polyhedral fans. In this way, the studied architecture may be applicable more broadly; however most immediate applications are in string theory. As a concrete example, dualGNN naturally extends to vex triangulations MacFadden & Sheridan (2025), which generate a strictly broader class of Calabi-Yau threefolds than the Batyrev construction. The vex setting is, combinatorially, simpler: its symmetry group is $\mathrm{GL}(d, \mathbb{Z})$ rather than $\mathrm{GL}(d, \mathbb{Z}) \ltimes \mathbb{Z}^d$, and its circuits satisfy a purely linear condition $\mathbf{A}\lambda = 0$ rather than the affine $[\mathbf{A}; \mathbf{1}]\lambda = 0$. dualGNN's circuit encoding handles this case directly, with the $\sum_i \lambda_i = 0$ constraint relaxed. The argument for such an application is identical to what we discuss here: by using a matroid to respect a problem's symmetries, one can obtain significantly stronger generalization and uniformity than direct sequence modeling.

To ease future comparison, the 20 held-out benchmark polygons, their exact FRT counts, and the uniformity-scoring protocol (collision counts and KL against the finite-sample noise floor) ship with the code in the repository's `eval/` directory, and dualGNN itself is available as an opt-in sampler in CYTools.

6 Acknowledgments

We would like to acknowledge the authors of Yip et al. (2025) for inspiring this project, and Jacky Yip in particular for providing the polytopes used in fig. 17. We would also like to acknowledge Jim Halverson, Liam McAllister, Andreas Schachner, and Elijah Sheridan for reading and providing feedback on this paper. Finally, I would like to acknowledge my wife Guin Gunter for her love and support.

This work was funded in part by NSF grant PHY-2309456.

Software development was assisted by Claude Opus 4.7 (Anthropic).

References

Gerardo Aldazabal, Luis E Ibáñez, Fernando Quevedo, and Angel M Uranga. D-branes at singularities: a bottom-up approach to the string embedding of the standard model. *Journal of High Energy Physics*,

2000(08):002–002, August 2000. ISSN 1029-8479. doi: 10.1088/1126-6708/2000/08/002. URL http://dx.doi.org/10.1088/1126-6708/2000/08/002.

Emile E. Anclin. An upper bound for the number of planar lattice triangulations. *Journal of Combinatorial Theory, Series A*, 103(2):383–386, 2003. ISSN 0097-3165. doi: https://doi.org/10.1016/S0097-3165(03)00097-9.

Victor V. Batyrev. Dual Polyhedra and Mirror Symmetry for Calabi-Yau Hypersurfaces in Toric Varieties. *J. Alg. Geom.*, 3:493–545, 1994.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Per Berglund, Giorgi Butbaia, Yang-Hui He, Elli Heyes, Edward Hirst, and Vishnu Jejjala. Generating triangulations and fibrations with reinforcement learning. *Physics Letters B*, 860:139158, 2025. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2024.139158. URL https://www.sciencedirect.com/science/article/pii/S0370269324007160.

Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Gunter M. Ziegler. *Oriented Matroids*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1999.

Ralph Blumenhagen, Volker Braun, Thomas W. Grimm, and Timo Weigand. GUTs in type IIB orientifold compactifications. *Nuclear Physics B*, 815(1-2):1–94, 2009. ISSN 0550-3213. doi: 10.1016/j.nuclphysb.2009.02.011. URL http://dx.doi.org/10.1016/j.nuclphysb.2009.02.011.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lió, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1026–1037. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/bodnar21a.html.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. URL https://arxiv.org/abs/2104.13478.

Aditi Chandra, Andrei Constantin, Cristofero S. Fraser-Taliente, Thomas R. Harvey, and Andre Lukas. Enumerating calabi-yau manifolds: Placing bounds on the number of diffeomorphism classes in the kreuzer-skarke list. *Fortschritte der Physik*, 72(5), March 2024. ISSN 1521-3978. doi: 10.1002/prop.202300264. URL http://dx.doi.org/10.1002/prop.202300264.

Mirjam Cvetič, Gary Shiu, and Angel M. Uranga. Three-family supersymmetric standardlike models from intersecting brane worlds. *Physical Review Letters*, 87(20), October 2001. ISSN 1079-7114. doi: 10.1103/physrevlett.87.201801. URL http://dx.doi.org/10.1103/PhysRevLett.87.201801.

Mirjam Cvetič, James Halverson, Ling Lin, Muyang Liu, and Jiahua Tian. Quadrillion F-theory compactifications with the exact chiral spectrum of the standard model. *Physical Review Letters*, 123(10), 2019. ISSN 1079-7114. doi: 10.1103/physrevlett.123.101601. URL http://dx.doi.org/10.1103/PhysRevLett.123.101601.

Jesús A. De Loera, Jörg Rambau, and Francisco Santos. *Triangulations*. Springer Berlin Heidelberg, 2010. ISBN 9783642129711. doi: 10.1007/978-3-642-12971-1. URL http://dx.doi.org/10.1007/978-3-642-12971-1.

Mehmet Demirtas, Liam McAllister, and Andres Rios-Tascon. Bounding the kreuzer-skarke landscape. *Fortschritte der Physik*, 68(11-12), October 2020. ISSN 1521-3978. doi: 10.1002/prop.202000086. URL http://dx.doi.org/10.1002/prop.202000086.

Mehmet Demirtas, Andres Rios-Tascon, and Liam McAllister. CYTools: A Software Package for Analyzing Calabi-Yau Manifolds. 11 2022.

Stefan Felsner and Lorenz Wernisch. Markov chains for linear extensions, the two-dimensional case. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '97, pp. 239–247, USA, 1997. Society for Industrial and Applied Mathematics. ISBN 0898713900.

Naomi Gendler, Nate MacFadden, Liam McAllister, Jakob Moritz, Richard Nally, Andreas Schachner, and Mike Stillman. Counting calabi-yau threefolds, 2023. URL https://arxiv.org/abs/2310.06820.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Emiel Hoogeboom, Jorn W.T. Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Kaibel, V. and Ziegler, G.M. *Counting Lattice Triangulations*, pp. 277–308. London Mathematical Society Lecture Note Series. Cambridge University Press, 2003.

Donald E Knuth. *Art of computer programming, volume 2*. Addison Wesley, Boston, MA, 3 edition, November 1997.

Maximilian Kreuzer and Harald Skarke. Complete classification of reflexive polyhedra in four dimensions. *Advances in Theoretical and Mathematical Physics*, 4(6):1209–1230, 2000. ISSN 1095-0753. doi: 10.4310/atmp.2000.v4.n6.a2. URL http://dx.doi.org/10.4310/ATMP.2000.v4.n6.a2.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Nate MacFadden. Efficient algorithm for generating homotopy inequivalent calabi-yaus, 2023. URL https://arxiv.org/abs/2309.10855.

Nate MacFadden. regfans, 2026. URL https://zenodo.org/doi/10.5281/zenodo.19406101.

Nate MacFadden and Elijah Sheridan. Calabi-yau threefolds from vex triangulations, 2025. URL https://arxiv.org/abs/2512.14817.

Nate MacFadden, Andreas Schachner, and Elijah Sheridan. The dna of calabi–yau hypersurfaces: A genetic algorithm for polytope triangulations. *Fortschritte der Physik*, 74(2), November 2025. ISSN 1521-3978. doi: 10.1002/prop.70060. URL http://dx.doi.org/10.1002/prop.70060.

Nate MacFadden, Stepan Yu. Orevkov, and Michael Stepniczka. Further bounding the kreuzer-skarke landscape, 2026. URL https://arxiv.org/abs/2602.16909.

Liam McAllister, Jakob Moritz, Richard Nally, and Andreas Schachner. Candidate de sitter vacua, 2024. URL https://arxiv.org/abs/2406.13751.

S. Meyer. Enumeration von triangulierungen. Diplomarbeit, Technische Universität Berlin, 2002. 87 pages.

S. Yu. Orevkov. Counting lattice triangulations: Fredholm equations in combinatorics. *Sb. Math.*, 213(11): 1530–1558, 2022. doi: 10.4213/sm9727e.

Jörg Rambau. TOPCOM: Triangulations of Point Configurations and Oriented Matroids. Technical Report 02-17, ZIB, Takustr. 7, 14195 Berlin, 2002.

E. Schönhardt. Über die zerlegung von dreieckspolyedern in tetraeder. *Mathematische Annalen*, 98(1): 309–312, March 1928. ISSN 1432-1807. doi: 10.1007/bf01451597. URL http://dx.doi.org/10.1007/BF01451597.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/e046ede63264b10130007afca077877f-Paper.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

Yiran Wang and Guido Montúfar. Trisearch: Learning to optimize triangulations via bistellar flips, 2026. URL https://arxiv.org/abs/2605.30220.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/bf00992696. URL http://dx.doi.org/10.1007/BF00992696.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Jacky H. T. Yip, Charles Arnal, François Charton, and Gary Shiu. Transforming calabi-yau constructions: Generating new calabi-yau manifolds with transformers, 2025. URL https://arxiv.org/abs/2507.03732.

A dualGNN Inference Visualization

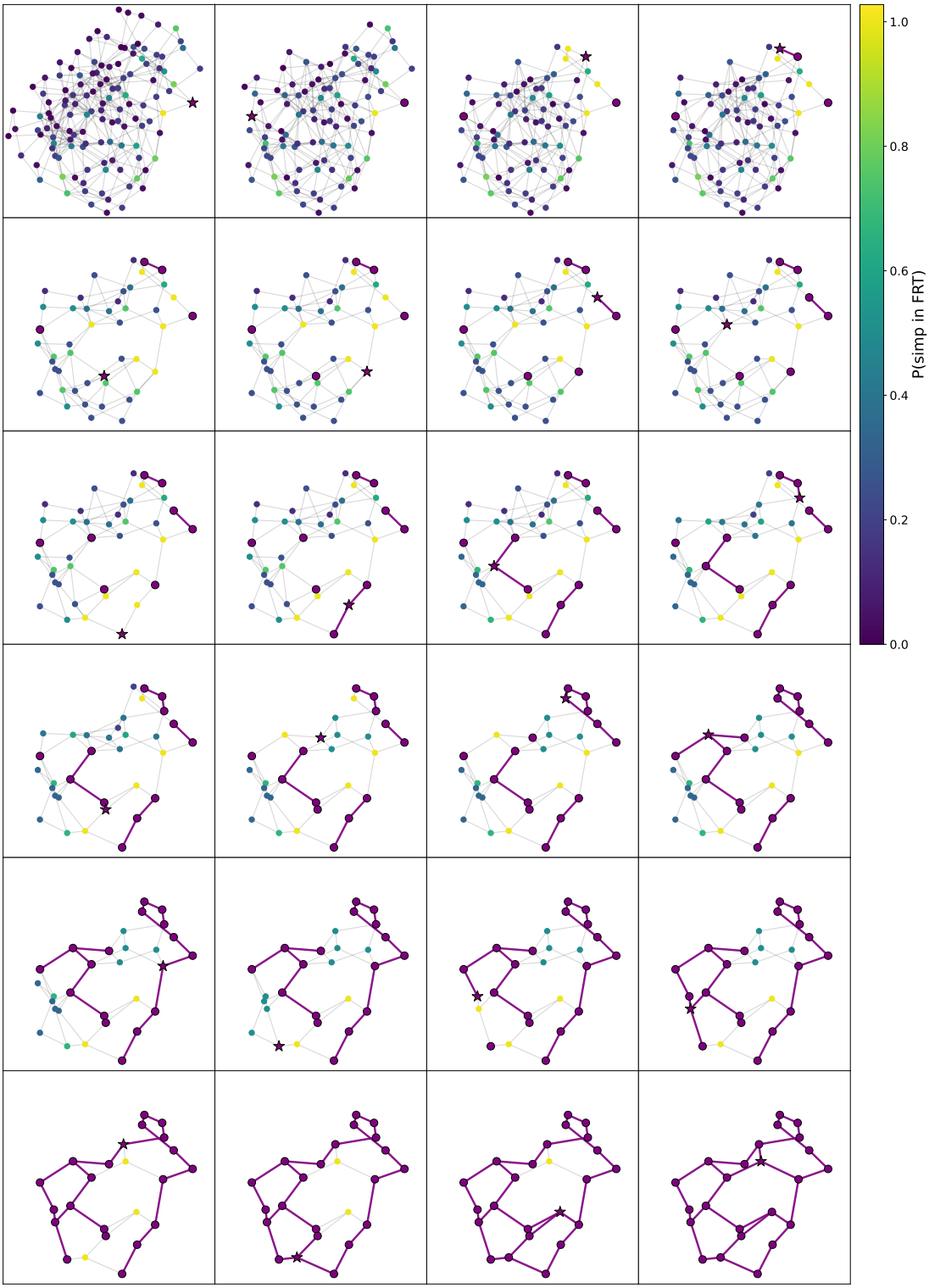

Figure 22: The inference rollout of dualGNN applied to the polygon in fig. 5. Nodes are colored with the probabilities dualGNN assigns them, with a purple star indicating the chosen simplex. Selected simplices are drawn in purple with edges between them also drawn in purple.

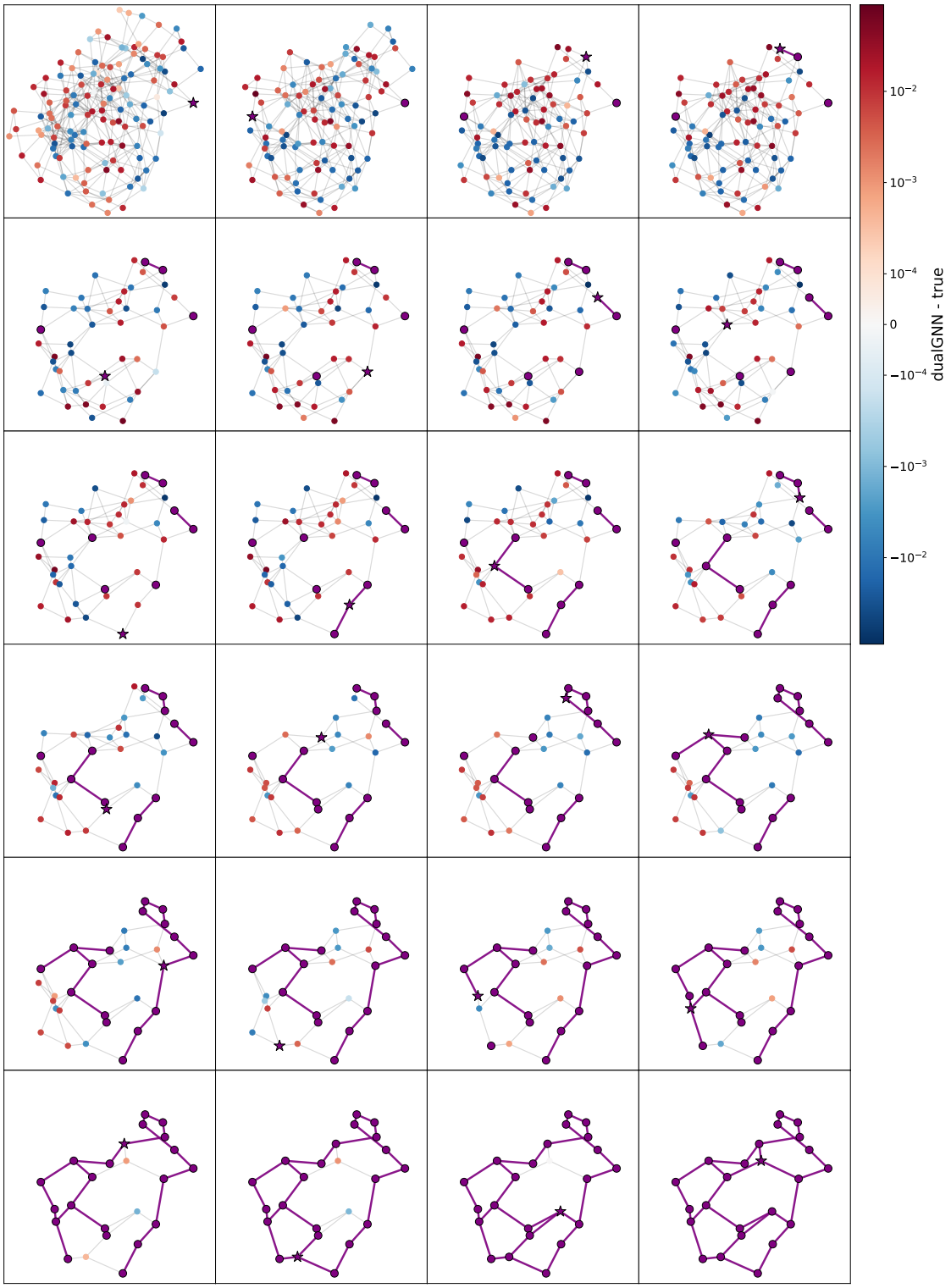

Figure 23: The inference rollout of dualGNN applied to the polygon in fig. 5. Nodes are colored with the difference in probability that dualGNN assigns them compared to the true probability computed from an exhaustive enumeration of FRTs. Selected simplices are drawn in purple with edges between them also drawn in purple.

B Classical Algorithms to Sample Triangulations

There are a large number of classical algorithms for randomly sampling fine triangulations. We provide a partial enumeration of algorithms in table 3, but do not claim this is complete.

Strategy	Procedure	Only regular?	Notes
`uniform`	Enumerate all triang. and then sample uniformly	No	Efficient for many samples or few triangulations
`flip_walk`	MCMC random walk on flip graph from seed triang.	No	See Demirtas et al. (2020). Also see Kaibel, V. and Ziegler, G.M. (2003) despite minor differences. Reported long mixing times
`grow2d`	Greedily add simplices to partial triangulation	No	See CYTools. Fast, semi-fair, optimized for 2D
`pushing`	Greedily 'place' points in the partial triangulation	Yes	See def 4.3.3 of De Loera et al. (2010). Misses some regular triangulations
`fast`[a]	Lift by heights sampled near a seed height vector	Yes	See Demirtas et al. (2020; 2022). Fast, local
`fair`[b]	Hybrid MCMC-like approach	Yes	See Demirtas et al. (2020; 2022). Not used in this work.

[a] Standing for `random_triangulations_fast` as in the CYTools implementation.
[b] Standing for `random_triangulations_fair` as in the CYTools implementation.

Table 3: Classical methods for sampling fine triangulations of lattice polygons.

Here we provide some of these algorithms, particularly those that we use as comparisons or those that have not appeared previously in the literature. First, we list the two from Demirtas et al. (2020) that we use in this work.

Algorithm 2: `random_triangulations_fast`

Input : Seed height vector $h_0 \in \mathbb{R}^{N_{\text{pts}}}$
Input : Standard deviation $c > 0$
Output: Triangulation $\mathcal{T}$ of Δ

Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, c^2\mathbf{I})$
$h \leftarrow h_0 + \boldsymbol{\epsilon}$
$\mathcal{T} \leftarrow \texttt{lift}(\Delta, h)$
return $\mathcal{T}$

Algorithm 3: `MCMC flip_walk`

Input : Seed triangulation $\mathcal{T}_0$ of Δ
Input : Number of steps k
Output: Triangulation $\mathcal{T}$ of Δ

$\mathcal{T} \leftarrow \mathcal{T}_0$
for $i \leftarrow 1$ **to** k **do**
  $\mathcal{N} \leftarrow \texttt{Neighbors}(\mathcal{T})$ `// triangulations reachable by one flip`
  $\mathcal{T} \leftarrow$ uniformly random element of $\mathcal{N}$
return $\mathcal{T}$

We also list the other two `grow2d` and `pushing` that we use heavily. The former was released with CYTools with MacFadden (2023); the latter is heavily inspired by TOPCOMRambau (2002). Both operate by adding simplices incrementally in a way that enables a fine triangulation (i.e., do not add simplices that cover

$> d+1$ lattice points for a d-dimensional polytope). These latter algorithms always converge in 2D but not more generally. `pushing` has one more guarantee: it always generates regular triangulations. This fact enables `pushing`, for the task of generating FRTs, to be, by far, the quickest algorithm. Not every regular triangulation is a pushing triangulation, though, so there are some FRTs that `pushing` cannot generate.

Algorithm 4: `grow2d`

Input : d-dimensional lattice polytope Δ with lattice points $\mathbf{A}$
Output: Fine triangulation $\mathcal{T}$ of Δ

$\sigma_0 \leftarrow$ `RandomSeedSimplex`(Δ)
$\mathcal{T} \leftarrow \{\sigma_0\}$
while $\mathcal{T}$ *does not triangulate* Δ **do**
  pick a facet f from `OpenFacets`$(\mathcal{T})$ `// ` $f \not\subset \partial\Delta$`, unmatched`
  $[x_{P(0)}, x_{P(1)}, \ldots] \leftarrow$ `RandomOrder`$(\mathbf{A})$
  foreach $x_{P(i)}$ *in order* **do**
    $\sigma \leftarrow f \cup \{x_{P(i)}\}$
    if $|\sigma \cap \mathbf{A}| = d+1$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{\sigma\}$
      break
return $\mathcal{T}$

Algorithm 5: `pushing`

Input : d-dimensional lattice polytope Δ with lattice points $\mathbf{A}$
Output: Fine triangulation $\mathcal{T}$ of Δ

$\sigma_0 \leftarrow$ `RandomSeedSimplex`(Δ)
$\mathcal{T} \leftarrow \{\sigma_0\}$
while $\mathcal{T}$ *does not triangulate* Δ **do**
  $\mathbf{A}_{\text{rem}} \leftarrow \{x \in \mathbf{A} : x \text{ not in any } \sigma \in \mathcal{T}\}$
  $[x_{P(0)}, x_{P(1)}, \ldots] \leftarrow$ `RandomOrder`$(\mathbf{A}_{\text{rem}})$
  foreach $x_{P(i)}$ *in order* **do**
    $\mathcal{S} \leftarrow \{f \cup \{x_{P(i)}\} : f \in$ `VisibleFacets`$(\mathcal{T}, x_{P(i)})\}$
    if $|\sigma \cap \mathbf{A}| = d+1$ *for every* $\sigma \in \mathcal{S}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{S}$
      break
return $\mathcal{T}$

C Transformer Baselines

As a baseline for dualGNN, inspired by Yip et al. (2025), we apply two transformer models to the task of generating FRTs of lattice polygons. These models differ slightly from CYTransformer: most notably, CYTransformer is encoder-decoder while, for simplicity, we study decoder-only models. Our two variants are similar (e.g., both use the standard attention mechanism Vaswani et al. (2017)), differing primarily in how lattice geometry is exposed to the model.

The first variant is the simpler of the two, using the serialization (semi-analogous to Yip et al. (2025))

```
x0,y0 x1,y1 ... | a0,b0,c0 a1,b1,c1 ....
```
(C.1)

The vocabulary here consists of

(a) the special characters `PAD`, `SPACE`, `EOS`, and delimiters (`,`, `|`) as well as

(b) a token for each integer $0, \ldots, C$ for C the maximum integer needed, as described below.

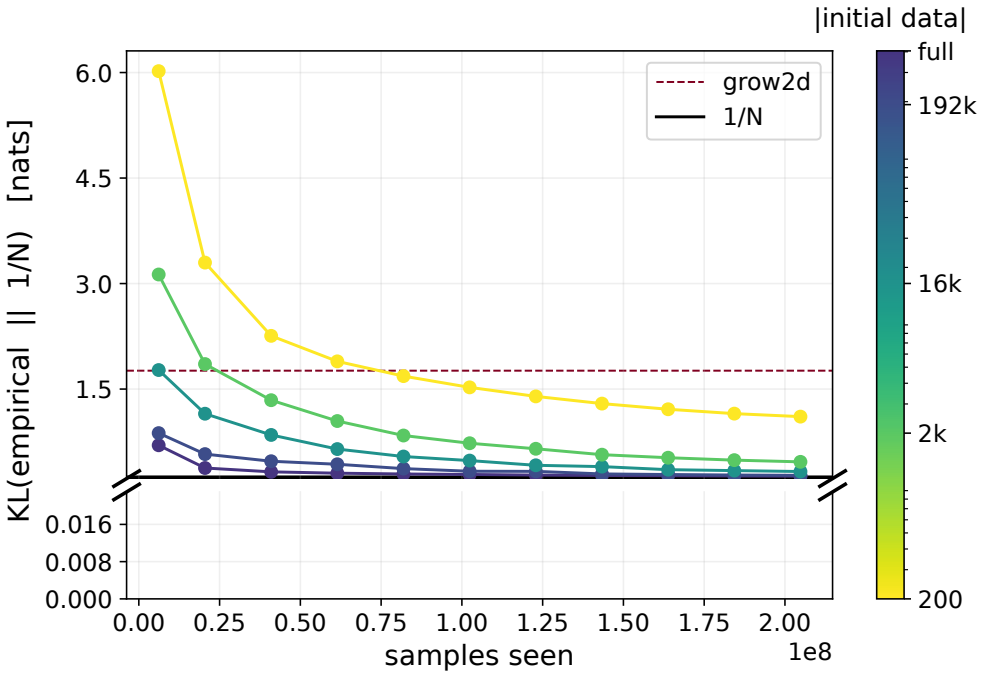

Figure 24: KL divergence to the uniform distribution over the course of training for the simple-encoding transformer on the polygon in fig. 5, at varying levels of initial training data scarcity. We only compare against `grow2d` since it generates samples quickly and somewhat uniformly.

Points are input coordinate-wise (x_i, y_i), translated so $\min_i(x_i) = \min_i(y_i) = 0$, while simplices are lists of point-indices (i, j, k). This means that C is the larger of the maximum coordinate value (across all x_i and y_i) and $N_{\text{pts}} - 1$. Positional information is supplied by a standard learned absolute positional embedding tied to sequence index. This serialization, in principle, can describe any polygon with $N_{\text{pts}} \leq C + 1$ and within the bounding box $[0, C]^2$, but we find it does not generalize strongly across polygons.

The second variant uses the same input/output stream, differing primarily in how point coordinates are exposed. Namely, points are directly input via their labels $0, \ldots, N_{\text{pts}} - 1$ with their coordinates (x_i, y_i) exposed via a variant of Rotary Position Embeddings Su et al. (2024) (RoPE). With `SPACE` no longer used to separate coordinate pairs, the sequences become more compact:

$$\texttt{012}\ldots\texttt{|}\,a_0 b_0 c_0 \ldots. \tag{C.2}$$

To encode the coordinates, we use a variant of the RoPE encoding for which we encode a 3D position (x_i, y_i, s) for s a 'simplex index', taking value $s = 0$ for the input points and $s = j + 1$ for the three tokens forming the jth simplex in the output. This is achieved by splitting each attention head's d_{head} dimensions into three chunks (d_x, d_y, d_s), with each chunk receiving a geometric-frequency rotation along its own coordinate. The `|` delimiter and special tokens (`PAD`, `EOS`) take sentinel position $(-1, -1, -1)$. The shared vocabulary makes this variant polygon-agnostic in principle.

We train both models on the polygon in fig. 5 at varying levels of initial training data and sample at temperature 1, allowing us to interpret the model's sampling behavior as the distribution it learned during training. Figures 24 and 25 show KL divergence to the uniform distribution as a function of training, for the simple and RoPE encodings respectively. To visualize the final trained RoPE model, we draw 10^6 samples and plot the frequency of each sampled triangulation versus its rank (in terms of highest frequency) in fig. 26.

These were the first models we studied (before even dualGNN). Both achieve strong single-polygon performance, especially the RoPE-variant which trains very quickly due to its compact serialization. These models are able to learn regularity, too — see fig. 27 where we apply a model trained on only 679 irregular trian-

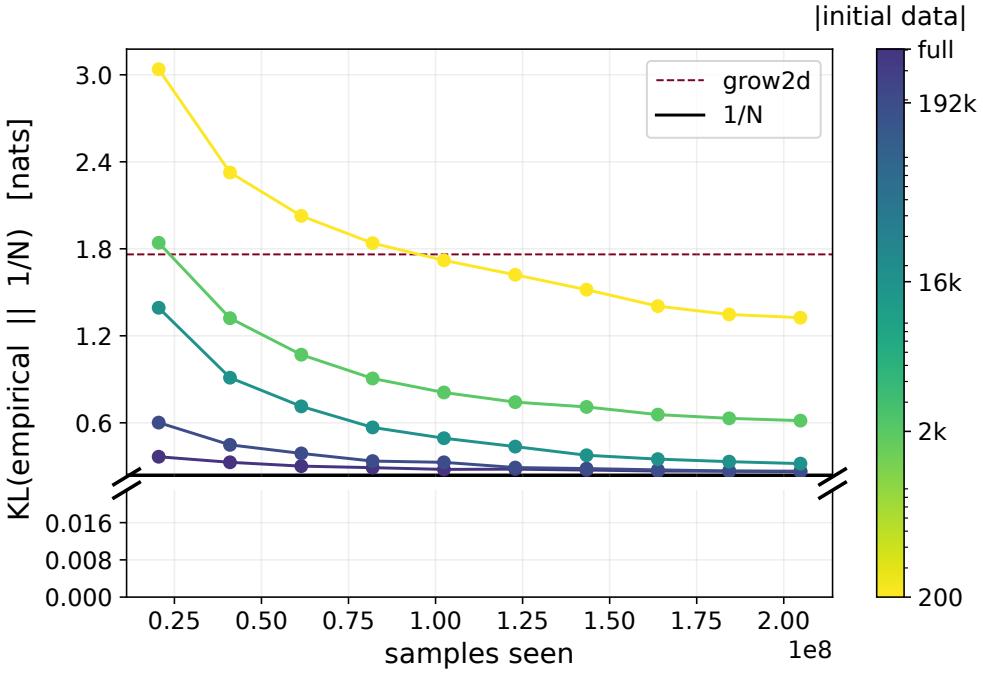

Figure 25: Same as fig. 24, but for the RoPE-encoded transformer.

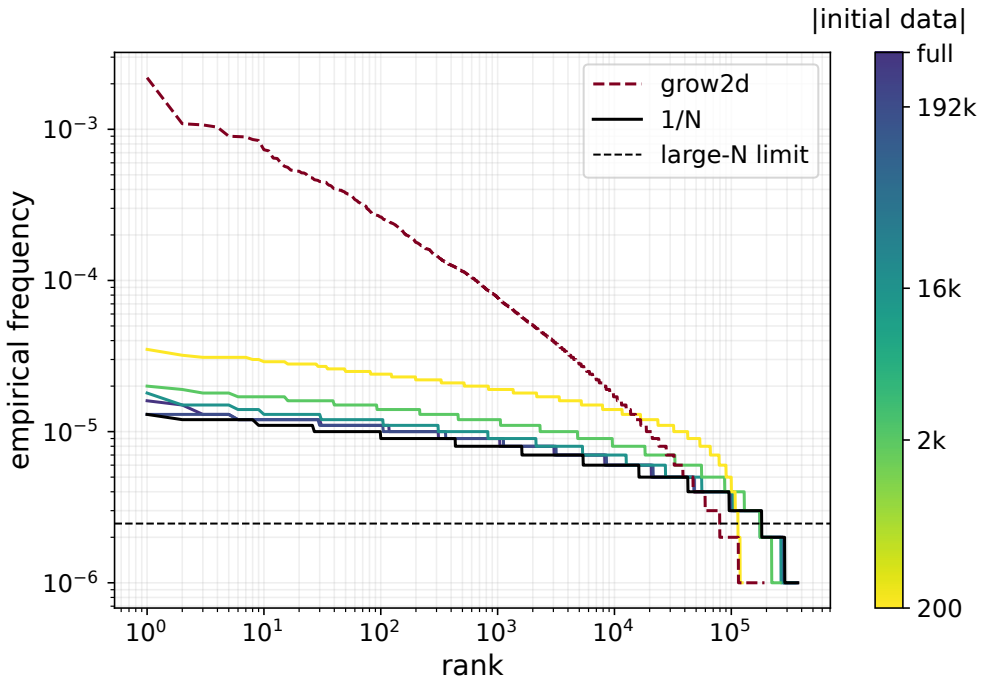

Figure 26: Rank-frequency plot of 10^6 samples drawn from the trained RoPE transformer on the polygon in fig. 5.

gulations of $[0,4]^2$ and compare its success in generating irregular triangulations to `grow2d`. We ultimately moved away from these models because they did not generalize across polygons, even when trained on many different ones simultaneously. We took this as motivation to better integrate the problem's symmetries into the architecture, leading to dualGNN.

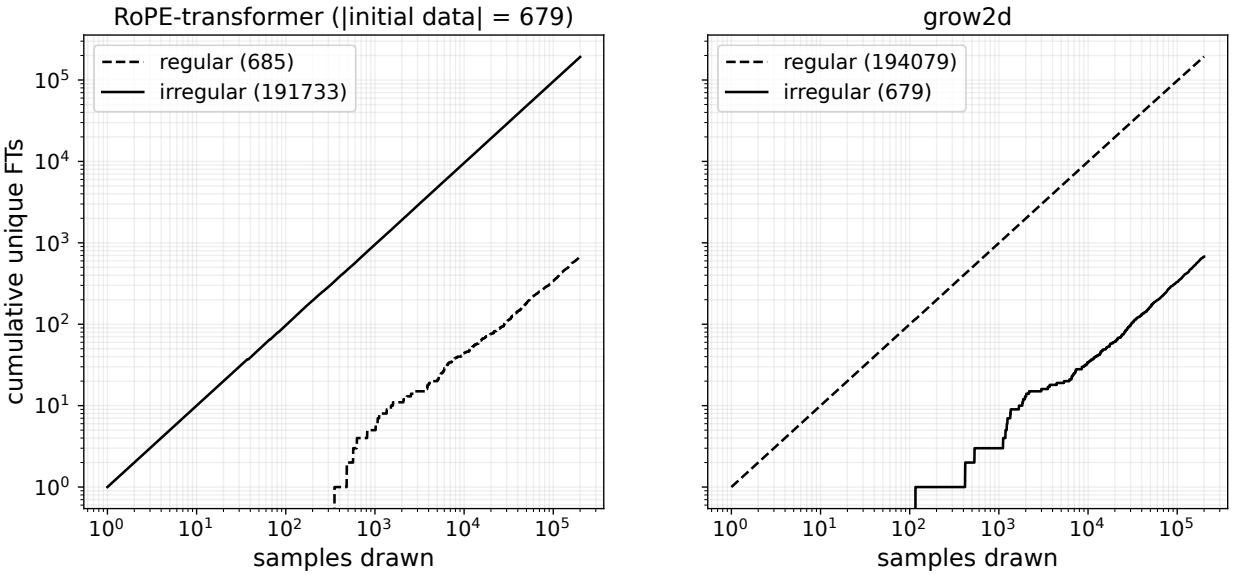

Figure 27: Number of regular and irregular triangulations generated for the polygon $[0,4]^2$ by the RoPE-based transformer trained on 679 irregular triangulations (left) and `grow2d` (right). The RoPE transformer learns to sample irregular triangulations effectively, with $> 95\%$ of the samples being unique irregular triangulations. Contrast this to `grow2d`, which generates only $\sim 0.34\%$ irregular triangulations, roughly matching the $\sim 0.2\%$ true fraction of irregular triangulations in $[0,4]^2$.