HyperGen: Learning Structure-Aware Spectral Flows for Hypergraph Generation

Yu Xie[1] **Junqi Liu**[1] **Hangyuan Du**[1] **Zhiqiang Wang**[1] **Ming Li**[2,*]
[1] MoE Key Lab of Computational Intelligence and Chinese Information Processing
Shanxi University, Taiyuan 030006, China
[2] Zhejiang Key Laboratory of Intelligent Education Technology and Application
Zhejiang Normal University, Jinhua 321004, China
{yuxie,liujunqi2,duhangyuan,wangzq}@sxu.edu.cn, mingli@zjnu.edu.cn

Abstract

Hypergraphs provide a natural representation for high-order relations, but real-world hypergraph data is often difficult to share or collect due to privacy, access, and annotation constraints. Hypergraph generation is therefore useful for constructing surrogate high-order relational data. Existing generators, however, typically build hyperedges through structural heuristics, propagation rules, or projected graph spaces, and thus do not directly model the incidence relation between nodes and hyperedges. Spectral representations offer a principled way to encode global and local relational patterns, but hypergraph spectra are intrinsically non-invertible: a spectral representation does not determine a unique discrete incidence structure. We propose HyperGen, a structure-aware generative framework that models hypergraph generation as continuous dynamics in a phase-enhanced spectral latent space. HyperGen constructs a Hermitian node–hyperedge spectral operator with learnable incidence phases, learns a conditional vector field that transports noise toward target spectral embeddings, and decodes the generated latents into node–hyperedge incidence matrices through a learned spectrum-to-structure mapping with denoising. Experiments on benchmark hypergraphs show that HyperGen more accurately recovers node–hyperedge incidences and better preserves spectral and geometric characteristics than existing baselines. Our code is available at `https://sxu-lau.github.io/HyperGen/`.

1 Introduction

Hypergraphs are powerful high-order data structures that extend traditional graphs by enabling hyperedges to connect multiple nodes simultaneously, thereby providing a richer representation of complex multi-way relationships. This capability allows hypergraphs to play an indispensable role across a broad range of real-world tasks, including drug discovery [1, 2, 3], computer vision [4, 5, 6, 7], traffic forecasting [8, 9], and molecular interaction modeling [10], where interactions are inherently multi-way. However, real-world hypergraph data is often limited by privacy, availability, and annotation costs. To address these challenges, hypergraph generative models have recently emerged as a promising approach for constructing high-quality high-order relational data in multi-way interaction scenarios [11, 12, 13].

Existing methods are largely structure-centric, as they directly generate hypergraph structures (i.e., hyperedges) through various mechanisms. HyperLap [14] generates hyperedges by hierarchically partitioning nodes and sampling within each group according to node degrees. HyperFF [15] models hypergraph evolution via a Forest Fire-based mechanism that jointly governs node- and

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

hyperedge-level propagation. DGMH [16] formulates hypergraph generation as sequential hyperedge construction in a latent space, where global structure emerges from shared node participation. HyperPLR [17] learns latent distributions on a projected weighted graph and generates hypergraphs by sampling cliques and performing clique covering. Consequently, these methods generally lack direct modeling of node–hyperedge interaction patterns in the incidence space, which makes it difficult to preserve fine-grained node–hyperedge dependencies and highlights the need for modeling such relationships more explicitly [18, 19, 20].

Inspired by spectral insights from graphs, we leverage the complementary roles of different frequency components—capturing global and local structural patterns—to guide the generative modeling of node–hyperedge relationships in the hypergraph spectral domain. However, despite their expressive power, spectral representations are difficult to exploit in practice due to the challenges of reconstructing hypergraphs from spectral data [21, 22]. In particular, standard hypergraph spectral operators—typically unoriented and unsigned—are inherently non-invertible, a difficulty rooted in fundamental limitations of spectral representations such as cospectral constructions and the limited recoverability of structure from spectra [18, 23, 24, 25, 26, 27]. This issue is further exacerbated in hypergraphs, where one spectral operator may correspond to multiple distinct hyperedge configurations [19, 28, 29].

To address the above limitations, we propose HyperGen, a hypergraph generative method that models hypergraph generation as a continuous process in a phase-enhanced spectral latent space. Without relying on explicit spectral inversion, HyperGen learns a structure-consistent mapping from spectral representations to node–hyperedge relations, enabling effective reconstruction of discrete hypergraph structures. Specifically, we introduce Peierls phases into the node–hyperedge incidence matrix to construct a phase-enhanced Hermitian spectral operator, which encodes discrete hypergraphs into a continuous latent space. We then model the generative process as a conditional vector field in the spectral space, capturing a continuous trajectory from noise to target spectral embeddings. Based on the generated spectral representations, we design a decoder that maps spectral latents into node–hyperedge incidence matrices. Furthermore, we apply a denoising process to obtain both the raw reconstructed hypergraph and its denoised version.

Our contributions are summarized as follows:

- **Spectral-to-incidence formulation.** We formulate hypergraph generation from the perspective of recovering node–hyperedge incidence structure from spectral latents, and identify spectral non-injectivity as the central obstacle that makes direct spectral inversion ill-posed.
- **Structure-aware spectral flow model.** We propose HyperGen, which constructs a phase-enhanced Hermitian hypergraph operator, encodes both low- and high-frequency spectral components, and learns a hypergraph-conditioned continuous vector field to generate structure-consistent spectral representations.
- **Learned spectrum-to-structure decoding.** We design a hyperedge decoder that maps generated spectral latents into node–hyperedge incidence structures, and combine amplitude prediction with denoising to suppress spurious relations in the reconstructed hypergraph.
- **Theoretical and empirical support.** We provide concrete theoretical results on spectral non-injectivity, phase-enhanced expressivity, and the well-posedness of the learned spectral flow, while experiments on real-world benchmark datasets demonstrate strong improvements in both incidence recovery and spectral consistency.

Due to space limitations, a detailed discussion of related work is deferred to Appendix A.

2 Preliminaries and Problem Setup

Hypergraph representation. A hypergraph is denoted by $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{C})$, where $\mathcal{V} = \{v_1, \dots, v_n\}$ is the node set, $\mathcal{E} = \{e_1, \dots, e_m\}$ is the hyperedge set, and $\mathbf{C} \in \{0,1\}^{n\times m}$ is the incidence matrix:

$$\mathbf{C}_{ij} = \begin{cases} 1, & v_i \in e_j, \\ 0, & v_i \notin e_j, \end{cases} \qquad i = 1, \dots, n,\ j = 1, \dots, m. \tag{1}$$

Each column $\mathbf{C}_{:j}$ is the indicator vector of hyperedge e_j, so a hyperedge is represented as an unordered subset of nodes and carries no internal order or orientation information. A weighted

hypergraph is written as $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{C}, w)$, where $w : \mathcal{E} \to \mathbb{R}_+$ assigns a non-negative weight to each hyperedge.

Generation objective. Given an observed hypergraph $\mathcal{H}$, the goal of hypergraph generation is to construct a synthetic hypergraph $\tilde{\mathcal{H}} = (\mathcal{V}, \tilde{\mathcal{E}}, \tilde{\mathbf{C}})$ that preserves the structural and incidence properties of the original hypergraph. In this paper, we focus on the fixed-node setting, where the node set is shared with the observed hypergraph and the generator produces an incidence matrix $\tilde{\mathbf{C}} \in \{0, 1\}^{n \times m}$ under a matched hyperedge budget. Formally, we seek $\tilde{\mathcal{H}}$ such that

$$\mathcal{S}(\tilde{\mathcal{H}}) \approx \mathcal{S}(\mathcal{H}), \tag{2}$$

where $\mathcal{S}(\cdot)$ denotes hypergraph statistics including hyperedge size distribution, node degrees, clustering coefficients, incidence patterns, and spectral characteristics. This formulation highlights that the task is not only to match coarse statistics, but also to recover node–hyperedge associations faithfully.

Spectral operator and the inversion challenge. Using $\mathbf{C}$ together with the hyperedge weight matrix $\mathbf{W} = \mathrm{diag}(w(e_1), \ldots, w(e_m))$, we define the normalized hypergraph Laplacian [30] as

$$\mathbf{L} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{C}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{C}^\top\mathbf{D}_v^{-1/2}, \tag{3}$$

where $\mathbf{D}_v = \mathrm{diag}(d(v_1), \ldots, d(v_n))$ and $\mathbf{D}_e = \mathrm{diag}(\delta(e_1), \ldots, \delta(e_m))$ are the node- and hyperedge-degree matrices. Because incidence columns encode unordered membership only, standard incidence-based spectral operators discard role or orientation information within each hyperedge. More importantly, the mapping from labeled hypergraphs to spectral representations is generally many-to-one: distinct incidence structures can induce the same spectral operator or the same spectrum. Consequently, recovering a unique incidence matrix from spectral information alone is ill-posed. Proposition 1 formalizes this point, which motivates learning a spectrum-to-structure decoder. For ease of reference, the notation used throughout the paper is summarized in Appendix B.

3 Method

We propose HyperGen, a hypergraph generative method whose pipeline is shown in Figure 1. It consists of a phase-aware hypergraph spectral encoder, a conditioned hypergraph spectral vector field in continuous latent space, and a hyperedge decoder with denoising.

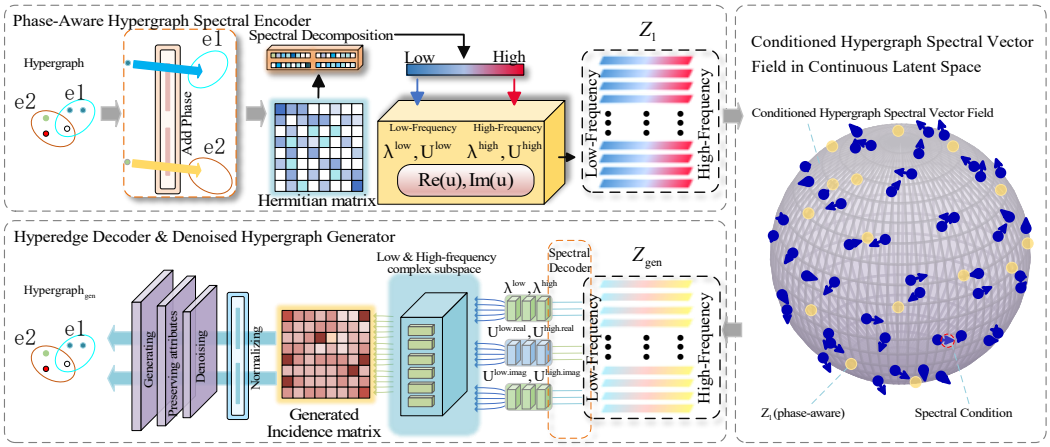

Figure 1: An overview of HyperGen, which consists of three key modules. The Phase-Aware Hypergraph Spectral Encoder (top-left) encodes the input hypergraph into a phase-aware spectral latent space (right). The Conditioned Hypergraph Spectral Vector Field in Continuous Latent Space (right) learns the continuous generation of high- and low-frequency spectral components from latent noise. Furthermore, the resulting spectral representations are fed into the Hyperedge Decoder & Denoised Hypergraph Generator (bottom-left), which reconstructs discrete hypergraphs.

3.1 Phase-Aware Hypergraph Spectral Encoder

To improve the expressiveness of hypergraph spectral representations, we introduce a phase-aware encoding mechanism that incorporates learnable phases into the node–hyperedge incidence matrix,

yielding a complex-valued representation that enriches spectral degrees of freedom beyond magnitude-only encoding. These spectral embeddings jointly encode node–hyperedge level structures, serving as the input for subsequent vector field modeling. Each entry of $\mathbf{C}$ is mapped to a complex-valued form as $\mathbf{C}_{ie}^{\text{phase}} = \mathbf{C}_{ie} \exp(\mathrm{i}\phi_{ie})$, where $\phi_{ie} \in \mathbb{R}$ is a learnable phase parameter. The resulting phase-augmented incidence matrix is $\mathbf{C}^{\text{phase}} \in \mathbb{C}^{n \times m}$, where $\mathbb{C}$ denotes the set of complex numbers. To enable joint spectral analysis of nodes and hyperedges, we convert the asymmetric node–hyperedge incidence relations into a Hermitian operator by embedding $\mathbf{C}^{\text{phase}}$ into a block-structured matrix

$$\mathbf{H} = \begin{pmatrix} \mathbf{0}_{n \times n} & \mathbf{C}^{\text{phase}} \\ (\mathbf{C}^{\text{phase}})^{\dagger} & \mathbf{0}_{m \times m} \end{pmatrix} \in \mathbb{C}^{(n+m) \times (n+m)}, \tag{4}$$

where $(\cdot)^{\dagger}$ denotes the conjugate transpose. We perform eigendecomposition of $\mathbf{H}$ as $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\dagger}$, where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_{n+m})$ contains eigenvalues in ascending order, and $\mathbf{U} \in \mathbb{C}^{(n+m) \times (n+m)}$ contains the corresponding orthonormal eigenvectors.

Low- and High-Frequency Spectral Encoding. The lowest k_{low} eigenpairs capture low-frequency components that encode global structural patterns in the hypergraph: $\mathbf{U}^{\text{low}} = \mathbf{U}_{[:,1:k_{\text{low}}]}, \quad \boldsymbol{\lambda}^{\text{low}} = (\lambda_1, \ldots, \lambda_{k_{\text{low}}})$. For each index $i \in \{1, \ldots, n+m\}$, the low-frequency spectral feature is constructed as $\mathbf{z}_i^{\text{low}} = \left[\boldsymbol{\lambda}^{\text{low}},\ \mathrm{Re}(\mathbf{U}_{i,:}^{\text{low}}),\ \mathrm{Im}(\mathbf{U}_{i,:}^{\text{low}})\right] \in \mathbb{R}^{3k_{\text{low}}}$. Similarly, the highest k_{high} eigenpairs are used to capture local structural patterns: $\mathbf{U}^{\text{high}} = \mathbf{U}_{[:,-k_{\text{high}}:]}, \quad \boldsymbol{\lambda}^{\text{high}} = (\lambda_{n+m-k_{\text{high}}+1}, \ldots, \lambda_{n+m})$. The high-frequency feature is defined as $\mathbf{z}_i^{\text{high}} = \left[\boldsymbol{\lambda}^{\text{high}},\ \mathrm{Re}(\mathbf{U}_{i,:}^{\text{high}}),\ \mathrm{Im}(\mathbf{U}_{i,:}^{\text{high}})\right] \in \mathbb{R}^{3k_{\text{high}}}$. The final phase-aware spectral embedding used as the target for vector field learning is obtained by concatenating the low- and high-frequency components:

$$\mathbf{Z}_1 = \left[\mathbf{Z}^{\text{low}} \parallel \mathbf{Z}^{\text{high}}\right] \in \mathbb{R}^{(n+m) \times (3k_{\text{low}} + 3k_{\text{high}})}. \tag{5}$$

3.2 Conditioned Hypergraph Spectral Vector Field in Continuous Latent Space

We instantiate the proposed spectral latent dynamics via a conditional vector field parameterized by a hypergraph convolutional network, operating on the spectral embeddings and conditioned on node–hyperedge incidence structure. Let $\mathbf{Z}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ denote an initial noise state. We construct a Gaussian probability path between $\mathbf{Z}_0$ and $\mathbf{Z}_1$:

$$\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t\mathbf{Z}_1 + \sigma\sqrt{t}\,\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{6}$$

where $t \in [0, 1]$. The conditional vector field $\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$ is defined on this trajectory and models the evolution of latent noise variables along it, conditioned on the phase-enhanced hypergraph structure, which is further constructed as follows. Specifically, we derive the phase-enhanced structure from $\mathbf{C}^{\text{phase}}$, and extract the node–hyperedge association set: $\mathcal{I} = \{(i, e) \mid |\mathbf{C}_{ie}^{\text{phase}}| > \varepsilon\}$ which is represented as $\mathbf{E} \in \mathbb{N}^{2 \times |\mathcal{I}|}$ with hyperedge weights $w \in \mathbb{R}^m$. For each $(i, e) \in \mathcal{I}$, we construct a local phase feature: $\mathbf{c}_{ie} = \big(\Re(\mathbf{C}_{ie}^{\text{phase}}), \Im(\mathbf{C}_{ie}^{\text{phase}})\big)$, where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary parts of a complex number, respectively. The feature $\mathbf{c}_{ie}$ is then encoded via ϕ_C and aggregated into a global context:

$$\mathbf{g}_C = \frac{1}{|\mathcal{I}|} \sum_{(i,e) \in \mathcal{I}} \phi_C(\mathbf{c}_{ie}). \tag{7}$$

To capture the temporal dynamics along the generative trajectory, temporal information is incorporated via a time embedding $\mathbf{e}_t = \phi_t(t)$, which is mapped to layer-wise conditioning terms via learnable functions: $\mathbf{s}_t^{(1)} = \phi_s^{(1)}(\mathbf{e}_t), \mathbf{s}_t^{(2)} = \phi_s^{(2)}(\mathbf{e}_t)$, where $\phi_s^{(1)}$ and $\phi_s^{(2)}$ are learnable mappings. The vector field $\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$ is implemented using stacked hypergraph convolution layers:

$$\mathbf{X}^{(1)} = \sigma(\mathrm{HGConv}_1(\mathbf{Z}_t, \mathbf{E}, w)) + \mathbf{s}_t^{(1)} + \mathbf{g}_C^{(1)}, \tag{8}$$

$$\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) = \mathrm{HGConv}_2(\mathbf{X}^{(1)}, \mathbf{E}, w) + \mathbf{s}_t^{(2)} + \mathbf{g}_C^{(2)}. \tag{9}$$

Training Objective. To guide noisy latent states toward the target spectral embedding, we define the regression target to supervise the vector field:

$$\mathbf{v}^\star(\mathbf{Z}_t, t) = \frac{\mathbf{Z}_1 - \mathbf{Z}_t}{1-t}, \quad t \in [0, 1). \tag{10}$$

We then learn a vector field that approximates this conditional expectation velocity field in the spectral space via a flow-matching objective:

$$\mathcal{L}_{\text{spec-flow}}(\psi) = \mathbb{E}_{t,\mathbf{Z}_t}\left[\|\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) - \mathbf{v}^\star(\mathbf{Z}_t, t)\|_2^2\right], \tag{11}$$

where the expectation is taken over the stochastic interpolation process induced by the Gaussian probability path introduced above, with $t \sim \mathcal{U}(0,1)$. At inference time, latent samples are generated by stochastically integrating the learned conditional vector field from Gaussian noise. We provide a theoretical analysis showing that phase-enhanced incidence improves the expressive capacity of hypergraph convolution and that the associated transport ODE admits a well-defined solution (see Appendices C.2 and C.3 for details).

3.3 Hyperedge Decoder & Denoised Hypergraph Generator

To alleviate the non-invertibility of spectrum-to-structure mappings, we design a hyperedge decoder that predicts node–hyperedge relations from latent representations generated by the conditional vector field, yielding the final hypergraph structure. Let $\mathbf{Z}_{\text{gen}} \in \mathbb{R}^{(n+m)\times d}$ denote the generated latent representations, where the first n rows correspond to nodes and the remaining m rows correspond to hyperedges, and d is the latent embedding dimension induced by the spectral factorization. The latent space is structured as a complex spectral factorization with low- and high-frequency components:

$$\mathbf{Z}_{\text{gen}} = \Big[\mathbf{\Lambda}^\ell, \Re(\mathbf{U}^\ell), \Im(\mathbf{U}^\ell), \mathbf{\Lambda}^h, \Re(\mathbf{U}^h), \Im(\mathbf{U}^h)\Big], \tag{12}$$

and the corresponding spectral basis is defined as $\mathbf{U} = [\mathbf{U}^\ell, \mathbf{U}^h]$. We extract hyperedge embeddings from the last m rows of $\mathbf{Z}_{\text{gen}}$, corresponding to indices $n+1$ to $n+m$. The e-th hyperedge embedding is given by $\mathbf{z}_e = \mathbf{Z}_{\text{gen},\, n+e}$. To model both structural and hyperedge-level weight information, a dual-decoder architecture consisting of an amplitude decoder and a structure decoder is constructed. The amplitude decoder predicts a normalized hyperedge weight: $\hat{a}_e = \sigma(f_{\text{amp}}(\mathbf{z}_e)) \in (0,1)$. Specifically, the supervision signal is derived from the projected graph by averaging intra-hyperedge connectivity and normalizing it across hyperedges: $\tilde{a}_e = \frac{\frac{1}{|\mathcal{V}_e|^2}\sum_{i,j\in\mathcal{V}_e}\mathbf{A}_{ij} - A_{\min}}{A_{\max} - A_{\min} + \epsilon}$, where $\mathbf{A}$ denotes the weighted adjacency matrix of the projected graph, $\mathcal{V}_e$ is the node set of hyperedge e. $A_{\min}$ and $A_{\max}$ are the minimum and maximum values of $\frac{1}{|V_e|^2}\sum_{i,j\in V_e}\mathbf{A}_{ij}$, respectively, and ϵ is a small constant for numerical stability. The amplitude decoder is optimized using an ℓ_2 regression objective: $\mathcal{L}_{\text{amp}} = \sum_{e=1}^{m} \|\hat{a}_e - \tilde{a}_e\|_2^2$.

We then decode $\mathbf{z}_e$ into a continuous node–hyperedge incidence matrix via a learnable structural decoder f_θ, which is trained as part of the conditioned hypergraph spectral vector field via velocity regression on a noisy interpolation trajectory: $\mathbf{C}_{\text{gen}}(:, e) = f_\theta(\mathbf{z}_e)$. To stabilize the decoded structure, the incidence magnitude is further normalized per hyperedge: $\tilde{\mathbf{C}}_{\text{gen}}(:, e) = \frac{|\mathbf{C}_{\text{gen}}(:,e)|}{\max_i |\mathbf{C}_{\text{gen}}(i,e)| + \varepsilon}$. And the predicted amplitude is mapped to the hyperedge weight range: $\hat{w}_e = \text{round}\big(\hat{a}_e(w_{\max} - w_{\min}) + w_{\min}\big)$. The weighted incidence matrix is obtained by modulating the decoded structure with the predicted hyperedge weights, followed by indicator thresholding to produce a discrete hypergraph:

$$\mathbf{C}_{\text{gen}}^{\text{prob}}(:, e) = \hat{w}_e \cdot \tilde{\mathbf{C}}_{\text{gen}}(:, e), \tilde{\mathbf{C}}_{\text{gen}}(i, e) = \mathbb{I}\big(\mathbf{C}_{\text{gen}}^{\text{prob}}(i, e) > \tau\big), \tag{13}$$

which yields the generated hyperedge set $\mathcal{E}_{\text{gen}} = \{\mathcal{V}_e^{\text{gen}}\}_{e=1}^m$, $\mathcal{V}_e^{\text{gen}} = \{i \mid \tilde{\mathbf{C}}_{\text{gen}}(i, e) = 1\}$, where $\mathbb{I}(\cdot)$ is the indicator function and τ is a threshold. We further obtain a denoised incidence matrix $\mathbf{C}_{\text{denoise}}$ by filtering low-confidence entries in $\mathbf{C}_{\text{gen}}^{\text{prob}}$, thereby reducing spurious connections introduced during decoding. This decoding connects spectral latents to discrete hypergraph structures. The complete training workflow and computational complexity analysis of HyperGen are provided in Appendix D.

3.4 Theoretical Insights

We summarize key theoretical insights that motivate the design of HyperGen; full proofs are deferred to Appendix C.

Proposition 1 (Spectral non-injectivity)**.** *Let $\Psi(\mathcal{H}) = \text{spec}(\mathbf{L}_{\mathcal{H}})$ denote the spectrum of the normalized incidence-based Laplacian of a labeled hypergraph. Then Ψ is not injective: there exist distinct labeled hypergraphs $\mathcal{H} \neq \mathcal{H}'$ such that $\Psi(\mathcal{H}) = \Psi(\mathcal{H}')$.*

Proposition 1 formalizes why spectra alone cannot identify a unique incidence structure and why HyperGen requires a learned spectrum-to-structure decoder rather than explicit spectral inversion.

Theorem 1 (Phase enhancement strictly generalizes unsigned incidence)**.** *Let* $\mathbf{C}^{\mathrm{phase}}_{ie} = \mathbf{C}_{ie}\exp(\mathrm{i}\phi_{ie})$ *and define the induced Hermitian operator*

$$\mathbf{H}(\mathbf{C}^{\mathrm{phase}}) = \begin{pmatrix} \mathbf{0} & \mathbf{C}^{\mathrm{phase}} \\ (\mathbf{C}^{\mathrm{phase}})^{\dagger} & \mathbf{0} \end{pmatrix}.$$

If $\phi_{ie} = 0$ *for all* (i, e)*, then* $\mathbf{H}(\mathbf{C}^{\mathrm{phase}})$ *reduces to the unsigned real-valued case. Moreover, if there exist two distinct nodes* $i \neq j$ *such that* $\sum_{e=1}^{m} \frac{\mathbf{C}_{ie}\mathbf{C}_{je}w(e)}{\delta(e)} \sin(\phi_{ie} - \phi_{je}) \neq 0$*, then the interaction term* $\left[\mathbf{C}^{\mathrm{phase}}\mathbf{W}\mathbf{D}_e^{-1}(\mathbf{C}^{\mathrm{phase}})^{\dagger}\right]_{ij}$ *is non-real.*

Theorem 1 shows that the phase-aware encoder strictly enlarges the set of distinguishable spectral patterns available to the vector field.

Theorem 2 (Well-posed spectral flow under flow matching)**.** *Assume the learned vector field* $\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$ *is continuous in* t*, locally Lipschitz in* $\mathbf{Z}$*, and has at most linear growth on* $t \in [0, 1]$*. Then the ODE*

$$\frac{d\mathbf{Z}_t}{dt} = \mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$$

admits a unique solution on $[0, 1]$*. Furthermore, for the noisy interpolation* $\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t\mathbf{Z}_1 + \sigma\sqrt{t}\,\varepsilon$*, we define the flow-matching regression target as*

$$\mathbf{v}^\star(\mathbf{Z}_t, t) = \frac{\mathbf{Z}_1 - \mathbf{Z}_t}{1-t}, \quad t \in [0, 1).$$

The regression target provides a reference for approximating the vector field at noisy latent states.

Theorem 2 establishes the well-posedness of the drift ODE underlying stochastic inference.

4 Experiments

In this section, we conduct comprehensive experimental evaluations to answer the following questions: **Q1**. To what extent can HyperGen accurately reconstruct the underlying incidence structure of real hypergraphs, as quantified by structural metrics? **Q2**. To what extent can HyperGen preserve the spectral and geometric characteristics of hypergraphs, as quantified by spectral and geometry metrics? **Q3**. How do different components of HyperGen affect its generation performance?

4.1 Experimental Settings

We compare **HyperGen** with representative hypergraph generation baselines on several benchmark hypergraph datasets, including HyperDK [31], HyperLap [14], HyperLap+ [14], THERA [32], HyperFF [15], and HyperPLR [17]. These methods cover different generation paradigms: HyperDK, HyperLap, and HyperLap+ are designed for static hypergraph generation; THERA targets temporal or dynamic hypergraphs; HyperFF models hypergraph evolution via a Forest Fire mechanism; and HyperPLR generates hypergraphs through a projected graph space. Following HyperPLR [17], we additionally evaluate two HyperDK variants to reduce sensitivity to parameter choices: HyperDK00, which preserves the average node degree and average hyperedge size, and HyperDK11, which preserves the degree of each node and the size of each hyperedge. All methods are evaluated under the same experimental protocol. Unless otherwise stated, all reported numerical results are rounded to two decimal places. Additional details are provided in Appendix E.

4.2 Structural Evaluation

We evaluate the generated hypergraphs using structural metrics, including Matched, Missing, Extra, Recall, Precision, and F1, which measure how accurately the generated hypergraphs recover the ground-truth node–hyperedge incidences. Detailed definitions of all evaluation metrics are provided in Appendix E.2. To answer **Q1**, we evaluate HyperGen on five benchmark datasets and compare its

Table 1: Structural performance comparison on temporal higher-order interaction networks.

Baseline	Dataset	Matched	Missing	Extra	Recall	Precision	F1
THERA	contact-high-school	1357	16835	19042	0.07	0.07	0.07
	contact-primary-school	3147	27582	32738	0.10	0.09	0.09
	email-Enron	118	4432	4822	0.03	0.02	0.02
	email-Eu	1166	84571	113991	0.01	0.01	0.01
	NDC-classes	14	6429	7198	0.00	0.00	0.00
HyperPLR	contact-high-school	6207	11985	14891	0.34	0.29	0.32
	contact-primary-school	6984	23745	34985	0.23	0.17	0.19
	email-Enron	1086	3464	3853	0.24	0.22	0.23
	email-Eu	7263	78474	71231	0.08	0.09	0.09
	NDC-classes	1820	4623	4233	0.28	0.30	0.29
HyperLap+	contact-high-school	896	17296	17296	0.05	0.05	0.05
	contact-primary-school	934	29795	31458	0.03	0.03	0.03
	email-Enron	82	4468	4486	0.02	0.02	0.02
	email-Eu	1175	84562	87606	0.01	0.01	0.01
	NDC-classes	13	6430	6455	0.00	0.00	0.00
HyperDK11	contact-high-school	8917	9275	92619	0.49	0.09	0.15
	contact-primary-school	12213	18516	139417	0.40	0.08	0.13
	email-Enron	883	3667	15449	0.19	0.05	0.08
	email-Eu	14376	71361	314419	0.17	0.04	0.07
	NDC-classes	68	6375	129517	0.01	0.00	0.00
HyperFF	contact-high-school	315	17877	3999	0.02	0.07	0.03
	contact-primary-school	518	30211	9649	0.02	0.05	0.03
	email-Enron	72	4478	1628	0.02	0.04	0.02
	email-Eu	602	85135	23888	0.01	0.02	0.01
	NDC-classes	12	6431	29871	0.00	0.00	0.00
HyperGen	contact-high-school	**18192**	**0**	2538294	**1.00**	0.01	0.01
	contact-primary-school	**30729**	**0**	7320768	**1.00**	0.00	0.01
	email-Enron	**4550**	**0**	211666	**1.00**	0.02	0.04
	email-Eu	**85737**	**0**	24891209	**1.00**	0.00	0.01
	NDC-classes	**6443**	**0**	1256725	**1.00**	0.01	0.01
HyperGen_DN	contact-high-school	**18192**	**0**	0	**1.00**	**1.00**	**1.00**
	contact-primary-school	**30729**	**0**	0	**1.00**	**1.00**	**1.00**
	email-Enron	**4550**	**0**	0	**1.00**	**1.00**	**1.00**
	email-Eu	**85737**	**0**	0	**1.00**	**1.00**	**1.00**
	NDC-classes	**6443**	**0**	0	**1.00**	**1.00**	**1.00**

raw (noisy) outputs and denoised version (HyperGen_DN) with several baseline methods. Results are reported in Table 1, and additional results (including error bars for Recall, Precision, and F1) for HyperGen are provided in Appendix F.1. The results of the variants HyperDK00 and HyperLap are deferred to Appendix F.2. As shown in Table 1, both HyperGen and HyperGen_DN achieve strong performance on structural matching metrics.

Moreover, different baseline methods exhibit distinct behaviors in structural matching. HyperPLR tends to achieve higher Recall but lower Precision, indicating over-generation behavior. In contrast, HyperFF generates hypergraphs that show limited consistency with the ground-truth structure and a reduced number of generated connections. On small-scale datasets such as contact-primary-school, several baselines (e.g., HyperDK11 and HyperLap+) exhibit instability with low numbers of matched node–hyperedge incidences. Notably, HyperGen_DN achieves near-perfect structural matching (Recall / Precision / F1 = 1.00) across all datasets, suggesting that HyperGen already generates structures closely aligned with the ground-truth distribution, and that denoising can fully recover exact incidences. Meanwhile, experiments at both the hypergraph and projected graph levels (see Appendix F.3) further demonstrate the effective generation capability of HyperGen.

4.3 Spectral Evaluation

To answer **Q2**, we investigate whether HyperGen preserves spectral and geometric properties in the latent space by evaluating Rel-Spec, TopK-Spec, Wasserstein distance, Frobenius norm, and

Table 2: Spectral and geometric comparison on temporal higher-order interaction networks.

Baseline	Dataset	Rel-Spec	TopK-Spec	Wasserstein	Frobenius	SCC
THERA	contact-high-school	0.13	86.12	16.45	768.39	5.95
	contact-primary-school	0.60	26.01	2.11	59.88	4.47
	email-Enron	0.28	159.11	30.07	616.91	5.71
	email-Eu	0.88	4k	448.90	8k	6.04
	NDC-classes	0.85	1k	109.65	1k	6.01
HyperPLR	contact-high-school	0.41	269.56	57.86	625.16	4.09
	contact-primary-school	34.69	1k	327.69	2k	5.90
	email-Enron	0.18	106.95	16.61	311.74	3.38
	email-Eu	0.34	1k	292.62	4k	5.44
	NDC-classes	0.22	314.07	37.56	760.86	4.06
HyperLap+	contact-high-school	0.10	68.72	14.12	768.49	6.03
	contact-primary-school	27.96	1k	266.68	2k	5.71
	email-Enron	0.20	118.72	19.40	620.68	5.76
	email-Eu	0.16	781.87	161.19	5k	6.01
	NDC-classes	0.65	906.12	95.16	1k	6.19
HyperDK11	contact-high-school	2.75	1k	403.35	4k	5.64
	contact-primary-school	51.28	2k	487.69	5k	6.09
	email-Enron	2.09	1k	233.33	1k	4.91
	email-Eu	1.21	5k	1k	13k	5.44
	NDC-classes	6.06	8k	1k	10k	5.81
HyperFF	contact-high-school	0.44	292.23	64.98	1k	6.01
	contact-primary-school	0.62	23.06	7.12	44.64	3.29
	email-Enron	0.64	368.71	63.69	573.91	5.81
	email-Eu	0.75	3k	715.23	5k	6.29
	NDC-classes	0.80	1k	213.11	2k	6.26
HyperGen	contact-high-school	3k	2556k	127k	2556k	6.04
	contact-primary-school	60.20	2k	257.01	2k	3.22
	email-Enron	378.90	215k	10k	216k	5.78
	email-Eu	5k	24975k	1249k	24976k	6.21
	NDC-classes	912.77	1262k	63k	1263k	6.25
HyperGen_DN	contact-high-school	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	contact-primary-school	**0.00**	**0.00**	**0.00**	**0.00**	**0.01**
	email-Enron	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	email-Eu	**0.00**	**0.00**	**0.01**	**0.00**	**0.00**
	NDC-classes	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

SCC, which measure eigenvalue deviation, spectral distribution discrepancy, and subspace alignment between generated and ground-truth hypergraphs. In particular, lower TopK-Spec and Wasserstein distances indicate that the spectral structure and eigenvalue distributions are closer to those of the ground truth, while reduced Frobenius norm and SCC further confirm stronger geometric consistency and subspace alignment with the ground truth. Here, "k" denotes values reported in thousands, rounded to the nearest integer. As shown in Table 2, the raw hypergraphs generated by HyperGen are affected by extra node–hyperedge relations, which introduce spectral perturbations. In contrast, the denoised version HyperGen_DN consistently reduces these distortions and achieves better spectral consistency across all datasets. Compared with other baselines, HyperFF generates fewer spurious connections; however, the relatively low proportion of ground-truth node–hyperedge incidences among all generated connections limits its overall performance in this setting.

In summary, after denoising, HyperGen preserves both spectral and geometric consistency, indicating that it can effectively generate the ground-truth node–hyperedge spectral distribution. Although the noise generated by HyperGen perturbs spectral measurements, the qualitative visualization in Appendix F.4 shows that the generated bipartite adjacency spectrum remains close to that of the ground truth.

4.4 Ablation Study

To answer **Q3**, we investigate the contribution of each component in HyperGen by conducting a comprehensive ablation study on the contact-high-school and email-Eu datasets. As shown in Table 3, we systematically consider several variants of the full framework, including w/o phase modeling (w/o phase), w/o high-frequency components (w/o HF), w/o low-frequency components (w/o LF), w/o noise in the training dynamics (w/o noise; i.e., setting the noise term in the training dynamics to zero), w/o projection-based hypergraph amplitude construction (w/o amp), and w/o the Hermitian hypergraph operator (w/o Herm).

Table 3: Ablation study on the contact-high-school and email-Eu datasets with structural matching and spectral consistency metrics under different model variants.

Method	Matched	Missing	Recall	RelSpec	TopKSpec	Wasserstein	SCC
contact-high-school							
w/o phase	17880	312	0.98	4k	2682k	134k	6.20
w/o HF	17639	553	0.97	4k	2721k	136k	6.22
w/o LF	17508	684	0.96	4k	2799k	140k	6.31
w/o noise	**18192**	**0**	**1.00**	**3k**	**2556k**	**127k**	**6.04**
w/o amp	17882	310	0.98	4k	2601k	137k	6.11
w/o Herm	17648	544	0.97	4k	2905k	155k	6.29
HyperGen	**18192**	**0**	**1.00**	**3k**	**2556k**	**127k**	**6.04**
email-Eu							
w/o phase	84814	923	0.99	6k	26395k	1319k	6.26
w/o HF	84234	1503	0.98	6k	26039k	1301k	6.24
w/o LF	83897	1840	0.98	6k	26809k	1339k	6.29
w/o noise	**85737**	**0**	**1.00**	**5k**	**24975k**	**1249k**	**6.21**
w/o amp	85280	457	0.99	6k	25976k	1298k	6.27
w/o Herm	84776	961	0.99	6k	26922k	1353k	6.30
HyperGen	**85737**	**0**	**1.00**	**5k**	**24975k**	**1249k**	**6.21**

We observe that, among the six ablated variants, all models except the w/o noise variant exhibit a clear performance degradation relative to HyperGen. In particular, these variants fail to faithfully preserve the node–hyperedge relationships of the original hypergraph and show consistently larger spectral discrepancies. Interestingly, the w/o noise variant, which removes stochasticity in the training dynamics by setting the noise term in the training interpolation path to zero, does not lead to a significant performance drop. This suggests that stochastic perturbation has limited impact on generation quality, and is consistent with the model primarily relying on the deterministic transport learned by the flow-matching objective. Additionally, we observe that the w/o HF variant performs better than the w/o LF variant, indicating that the model relies more heavily on low-frequency spectral components, which have a more significant impact on overall performance. Overall, the above results demonstrate that each component in HyperGen plays a role in shaping its generation performance.

5 Conclusion

We present HyperGen, a structure-aware hypergraph generation framework based on continuous spectral flows. Instead of attempting to invert hypergraph spectra directly, HyperGen learns a phase-enhanced spectral latent representation and a decoder that maps generated latents into node–hyperedge incidence structures. This design addresses the non-invertibility of standard spectral operators while preserving the relational information needed for discrete hypergraph generation. Experiments on multiple benchmark datasets show that HyperGen improves incidence recovery and maintains stronger spectral and geometric consistency than existing baselines, with denoising further reducing spurious node–hyperedge relations. Technically and theoretically, our work suggests that spectral latent dynamics provide a useful bridge between continuous generative modeling and discrete high-order structure recovery. Future work can focus on improving scalability through approximate eigensolvers and lightweight vector fields, extending the framework to variable-size hypergraph generation, and studying stronger privacy-aware generation objectives for sensitive hypergraph data.

6 Limitations

The main limitation of HyperGen is its reliance on eigendecomposition to compute phase-aware spectral embeddings. Although HyperGen achieves strong generative performance across the benchmarks, the associated spectral computations incur increasing computational costs and memory consumption as the scale of the input hypergraph increases, potentially making HyperGen less practical for large-scale hypergraphs. Improving the model's scalability for generation remains a direction for future work.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (Nos. 62472270, 62622615, 62576198), and the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2024C03262).

We declare no competing interests.

References

[1] Xiaobao Huang, Yihong Ma, Anjali Gurajapu, Jules Schleinitz, Zhichun Guo, Ting Hua, Sarah E Reisman, and Nitesh V Chawla. ChemHGNN: A hierarchical hypergraph neural network for reaction virtual screening and discovery. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 11108–11119, 2026.

[2] Lu Shen, Feng Hu, and Libing Bai. A multi-layer hypergraph framework for drug-drug interaction prediction based on transformer and hypergraph convolution. *Computational Biology and Chemistry*, 122: 108880, 2026.

[3] Maryam Mehrabani, Amir Lakizadeh, Alireza Fotuhi Siahpirani, Mahdieh Salimi, Fatemeh Zare-Mirakabad, and Ali Masoudi-Nejad. SynergyGraph: Predicting cell line specific drug combination synergy scores using knowledge graph representation and hypergraph modeling. *Scientific Reports*, 16: 3131, 2026.

[4] Joshua Fixelle. Hypergraph vision transformers: Images are more than nodes, more than edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9751–9761, 2025.

[5] Lin Bie, Siqi Li, Yifan Feng, and Yue Gao. Hyper-Depth: Hypergraph-based multi-scale representation fusion for monocular depth estimation. In *Proceedings of the 20th IEEE/CVF International Conference on Computer Vision*, pages 5081–5090, 2025.

[6] Hao Wang, Shuo Zhang, and Biao Leng. HGFormer: Topology-aware vision transformer with hypergraph learning. *IEEE Transactions on Multimedia*, 27:5746–5757, 2025.

[7] Caoshuo Li, Tanzhe Li, Xiaobin Hu, Donghao Luo, and Taisong Jin. DVHGNN: Multi-scale dilated vision HGNN for efficient vision recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20158–20168, 2025.

[8] Yeqing Lin, Lai Wei, and Longfei Hu. Hypergraph-based multi-scale spatio-temporal graph convolution network for traffic forecasting. *International Journal of Machine Learning and Cybernetics*, 17(4):162, 2026.

[9] Minze Li, Feng Shao, Pengjie Liu, Wenhao Liu, Hu Shao, and Liang Shen. Multi-hypergraph fusion for network-wide traffic flow prediction: A spatiotemporal hypergraph neural network. *Knowledge-Based Systems*, page 116083, 2026.

[10] Bowen Wang, Junyou Li, Donghao Zhou, Lanqing Li, Jinpeng Li, Ercheng Wang, Jianye Hao, Liang Shi, Chengqiang Lu, Jiezhong Qiu, et al. Unified and explainable molecular representation learning for imperfectly annotated data from the hypergraph view. *Nature Communications*, 16:8717, 2025.

[11] Quintino Francesco Lotito, Federico Musciotto, Alberto Montresor, and Federico Battiston. Higher-order motif analysis in hypergraphs. *Communications Physics*, 5(1):79, 2022.

[12] Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. Structural patterns and generative models of real-world hypergraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 176–186, 2020.

[13] Dorian Gailhard, Enzo Tartaglione, Lirida Naviner, and Jhony H. Giraldo. HYGENE: A diffusion-based hypergraph generation method. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pages 16682–16690, 2025.

[14] Geon Lee, Minyoung Choe, and Kijung Shin. How do hyperedges overlap in real-world hypergraphs? - patterns, measures, and generators. In *Proceedings of the 30th World Wide Web Conference*, pages 3396–3407, 2021.

[15] Yunbum Kook, Jihoon Ko, and Kijung Shin. Evolution of real-world hypergraphs: Patterns and models without oracles. In *Proceedings of the 20th IEEE International Conference on Data Mining*, pages 272–281, 2020.

[16] Xianglin Zuo, Wenqi Chen, Xianduo Song, Xin Wang, and Ying Wang. Generating real-world hypergraphs via deep generative models. *Information Sciences*, 647:119412, 2023.

[17] Weihuang Wen and Tianshu Yu. HyperPLR: Hypergraph generation through projection, learning, and reconstruction. In *Proceedings of the 13th International Conference on Learning Representations*, pages 84220–84236, 2025.

[18] T-H Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph Laplacian and approximation algorithms. *Journal of the ACM*, 65(3):1–48, 2018.

[19] Shashwath S Shetty and K. Arathi Bhat. Spectral theory of hypergraphs: A survey. *arXiv preprint arXiv:2507.13664*, 2025.

[20] Uthsav Chitra and Benjamin Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1172–1181, 2019.

[21] Dragoš M. Cvetković, Michael Doob, and Horst Sachs. *Spectra of Graphs: Theory and Application*. 1980.

[22] Anand Louis. Hypergraph Markov operators, eigenvalues and approximation algorithms. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 713–722, 2015.

[23] Lothar Collatz and Ulrich Sinogowitz. Spektren endlicher grafen. *Abh. Math. Semin. Univ. Hambg.*, 21: 63–77, 1957.

[24] Luke Duttweiler and Nathan Reff. Spectra of cycle and path families of oriented hypergraphs. *Linear Algebra and its Applications*, 578:251–271, 2019.

[25] Li Liang, Shi-Ming Cai, and Shi-Cai Gong. Hypergraph representation learning with weighted- and clustering-biased random walks. *Entropy*, 27(10):1072, 2025.

[26] Samiron Parui. On the incidence matrices of hypergraphs. *Linear and Multilinear Algebra*, 73(17): 3861–3880, 2025.

[27] Nathan Reff. Spectral properties of oriented hypergraphs. *Electronic Journal of Linear Algebra*, 27: 373–391, 2014.

[28] Joshua Cooper and Aaron Dutle. Spectra of uniform hypergraphs. *Linear Algebra and its Applications*, 436(9):3268–3292, 2012.

[29] Yanbang Wang and Jon Kleinberg. From graphs to hypergraphs: Hypergraph projection and its reconstruction. In *Proceedings of the 12th International Conference on Learning Representations*, pages 52745–52775, 2024.

[30] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pages 1601–1608, 2006.

[31] Kazuki Nakajima, Kazuyuki Shudo, and Naoki Masuda. Randomizing hypergraphs preserving degree correlation and local clustering. *IEEE Transactions on Network Science and Engineering*, 9(3):1139–1153, 2022.

[32] Sunwoo Kim, Fanchen Bu, Minyoung Choe, Jaemin Yoo, and Kijung Shin. How transitive are real-world group interactions? - measurement and reproduction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1132–1143, 2023.

[33] Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, pages 61963–61999, 2024.

[34] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, pages 2672–2680, 2014.

[35] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 610–619, 2018.

[36] Yuwei Xu and Chenhao Ma. MoDiff - graph generation with motif-aware diffusion model. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3437–3448, 2025.

[37] Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15159–15179, 2022.

[38] Tianze Luo, Zhanfeng Mo, and Sinno Jialin Pan. Fast graph generation via spectral diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3496–3508, 2024.

[39] Dorian Gailhard, Enzo Tartaglione, Lirida Naviner, and Jhony H. Giraldo. Feature-aware (hyper)graph generation via next-scale prediction. In *Proceedings of the 43rd International Conference on Machine Learning*, pages 32640–32674, 2026.

[40] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48): E11221–E11230, 2018.

Appendix

The appendix is organized as follows: Appendix A provides related works. Appendix B summarizes the notation used in this paper. Appendix C presents proofs for theoretical insights. Appendix D provides the training algorithm and time complexity analysis of HyperGen. Appendix E describes the details of experimental settings. Finally, Appendix F presents supplementary experiments.

A Related Work

A.1 Graph Generation Methods

Graph generation in the pairwise-graph setting has been extensively studied and provides useful methodological context for hypergraph generation. Existing approaches can be broadly organized into three lines of work. Deep learning–based methods form a major family of graph generators. LGD [33] enables end-to-end graph generation by learning latent graph distributions. More broadly, many neural generative models aim to capture the underlying graph distribution directly. Within this line, adversarial methods originating from GANs [34] have also been adapted to graph data. NetGAN [35], for example, employs adversarial training over random walks to model graph connectivity patterns and improve the realism of generated graphs. Another line of work uses recurrent subgraph motifs as explicit generative primitives. MoDiff [36] incorporates motif-aware diffusion to model local motif structures in temporal graphs, allowing generated samples to better match the source graphs in terms of motif distributions and evolutionary behavior. A third line of work leverages spectral information to capture both global topology and local structural variation. SPECTRE [37] conditions generation on dominant components of the graph Laplacian spectrum and reconstructs graphs that are consistent with the corresponding eigenvalues and eigenvectors. GSDM [38] further models graph generation in spectral space via low-rank diffusion stochastic differential equations, producing graphs that better preserve both global topology and local structure. These studies demonstrate the value of spectral representations for generative modeling. However, extending spectral graph generators to hypergraphs is non-trivial, since hypergraphs are defined by node–hyperedge incidence and high-order interactions rather than pairwise edges alone.

A.2 Hypergraph Generation Models

Compared with graph generation, generative modeling for hypergraphs remains relatively underexplored. Existing methods can be organized into three broad categories. A prominent line of work generates hypergraphs by explicitly reproducing structural regularities observed in data. HyperDK [31] randomizes hypergraphs while preserving degree correlation and local clustering, thereby maintaining important combinatorial and connectivity statistics. HyperLap [14] and HyperLap+ generate hypergraphs by modeling empirical overlap patterns among hyperedges. THERA [32] focuses on reproducing high-order group connectivity by exploiting community structure to capture transitivity patterns observed in real hypergraphs. HyperFF [15] models hypergraph evolution through a Forest Fire-style propagation process that jointly governs node- and hyperedge-level growth. DGMH [16] formulates hypergraph generation as sequential hyperedge construction in a latent space, allowing global structure to emerge through shared node participation. HyperPLR [17] takes a different route by projecting a hypergraph into a weighted graph, learning latent representations in the projected space, and reconstructing hyperedges through a unified projection–learning–reconstruction pipeline. Another category constructs hypergraphs through stochastic or rule-driven mechanisms. HYGENE [13] adopts a diffusion-based local expansion strategy on the bipartite representation of a hypergraph, iteratively adding nodes and hyperedges and progressively refining both global organization and local detail through denoising diffusion. Recent work has also begun to consider joint generation of topology and attributes. FAHNES [39], for instance, is a feature-aware framework that models hypergraph structure and node features jointly through hierarchical next-scale prediction, thereby moving beyond purely topology-only generation.

Overall, existing hypergraph generators have made encouraging progress in matching coarse structural statistics and improving the diversity and fidelity of synthesized hypergraphs. Nevertheless, many of these methods remain structure-centric: they rely on heuristic hyperedge construction, propagation rules, or projected graph spaces, and are therefore not designed to model fine-grained node–hyperedge

dependencies directly in the incidence space. As a result, jointly preserving incidence structure and spectral consistency remains challenging.

A.3 Positioning of HyperGen

HyperGen is related to both spectral graph generation and hypergraph structure generation, but it differs from each line in a fundamental way. Relative to spectral graph generators such as SPECTRE [37] and GSDM [38], HyperGen operates on hypergraph-specific node–hyperedge representations rather than pairwise graph spectra. Relative to existing hypergraph generators such as HyperLap [14], HyperFF [15], DGMH [16], and HyperPLR [17], HyperGen does not generate hyperedges solely through heuristic structural rules, sequential construction, or projected-graph reconstruction. Instead, it learns continuous latent dynamics in a phase-enhanced spectral space and decodes the resulting representations into node–hyperedge incidence structures through a learned spectrum-to-structure mapping. This distinction is important because standard hypergraph spectra are generally non-invertible; accordingly, HyperGen should not be interpreted as a direct spectral inversion method, but rather as a structure-aware generative framework that combines spectral modeling with explicit incidence recovery.

B Notation

For ease of reference, Table 4 summarizes the key notation used throughout the paper, especially the symbols that appear most frequently in the main text, where boldface is omitted for simplicity.

Table 4: Key notation used in the main text.

Symbol	Description
$V = \{v_i\}_{i=1}^{n},\ E = \{e_j\}_{j=1}^{m}$	Node set and hyperedge set, where $n = \lvert V\rvert$ and $m = \lvert E\rvert$.
$C \in \{0,1\}^{n\times m}$	Binary node–hyperedge incidence matrix.
$w,\ W$	Hyperedge weight function $w : \mathcal{E} \to \mathbb{R}_+$ and diagonal weight matrix $W = \mathrm{diag}(w(e_1), \ldots, w(e_m))$.
$D_v,\ D_e$	Diagonal matrices of node degrees and hyperedge degrees, respectively.
L	Normalized hypergraph Laplacian, $L = I - D_v^{-1/2} C W D_e^{-1} C^\top D_v^{-1/2}$.
ϕ_{ie}	Learnable phase associated with node–hyperedge pair (i, e).
C^{phase}	Phase-augmented complex incidence matrix, $C^{\mathrm{phase}}_{ie} = C_{ie}\exp(\mathrm{i}\phi_{ie})$.
H	Block-Hermitian spectral operator constructed from C^{phase}.
$U,\ \Lambda$	Eigenvector matrix and diagonal eigenvalue matrix in the decomposition $H = U\Lambda U^\dagger$.
$k_{\mathrm{low}},\ k_{\mathrm{high}}$	Numbers of retained low-frequency and high-frequency eigenpairs.
$U_{\mathrm{low}},\ U_{\mathrm{high}},\ \lambda_{\mathrm{low}},\ \lambda_{\mathrm{high}}$	Truncated eigenvectors and eigenvalues corresponding to the low- and high-frequency spectral components.
Z_1	Target phase-aware spectral embedding formed by concatenating low- and high-frequency features.
$Z_0,\ Z_t$	Initial Gaussian latent state and intermediate latent state along the probability path.
$t,\ \epsilon,\ \sigma$	Time variable, Gaussian perturbation, and noise scale in the interpolation $Z_t = (1-t)Z_0 + tZ_1 + \sigma\sqrt{t}\,\epsilon$.
$v_\psi(Z_t, t),\ v^\star(Z_t, t),\ L_{\text{spec-flow}}$	Learned conditional vector field, closed-form target velocity, and spectral flow-matching loss.
$Z_{\mathrm{gen}},\ z_e$	Generated latent representation and the latent embedding of hyperedge e.
$\hat{a}_e,\ \tilde{a}_e,\ L_{\mathrm{amp}}$	Predicted hyperedge amplitude, target amplitude, and amplitude regression loss.
$C_{\mathrm{gen}},\ \tilde{C}_{\mathrm{gen}}$	Decoded continuous incidence matrix and its per-hyperedge normalized version.
$C^{\mathrm{prob}}_{\mathrm{gen}},\ C_{\mathrm{denoise}},\ \hat{w}_e$	Weighted/probabilistic incidence matrix, denoised incidence matrix, and decoded hyperedge weight.

C Proofs for Theoretical Insights

C.1 Restatement and Proof of Proposition 1

We formally prove that standard hypergraph spectral operators based on the incidence matrix are non-injective, which mathematically implies that hypergraph generation from spectral representations is ill-posed.

Proposition 1 (Spectral non-injectivity)**.** *Let $\Psi(\mathcal{H}) = \mathrm{spec}(\mathbf{L}_{\mathcal{H}})$ denote the spectrum of the normalized incidence-based Laplacian of a labeled hypergraph. Then Ψ is not injective: there exist distinct labeled hypergraphs $\mathcal{H} \neq \mathcal{H}'$ such that $\Psi(\mathcal{H}) = \Psi(\mathcal{H}')$.*

Proof. Due to the unoriented nature of the incidence matrix $\mathbf{C}$, permutations of vertices within a hyperedge leave its corresponding incidence column invariant:

$$\mathbf{C}'_{:j} = \mathbf{P}_\pi \mathbf{C}_{:j} = \mathbf{C}_{:j}, \tag{14}$$

where π is any permutation of vertices in hyperedge e_j and $\mathbf{P}_\pi$ is the corresponding permutation matrix. We choose a nontrivial permutation such that $\mathbf{C}' = \mathbf{P}_\pi \mathbf{C} \neq \mathbf{C}$. Extending this to the entire incidence matrix, we have:

$$\mathbf{C}' = \mathbf{P}_\pi \mathbf{C}, \quad \mathbf{D}'_v = \mathbf{P}_\pi \mathbf{D}_v \mathbf{P}_\pi^\top, \quad \mathbf{W}' = \mathbf{W}, \quad \mathbf{D}'_e = \mathbf{D}_e. \tag{15}$$

Substituting into the normalized Laplacian definition gives:

$$\begin{aligned} \mathbf{L}' &= \mathbf{I} - (\mathbf{D}'_v)^{-1/2} \mathbf{C}' \mathbf{W}' \mathbf{D}'^{-1}_e (\mathbf{C}')^\top (\mathbf{D}'_v)^{-1/2} \\ &= \mathbf{I} - \mathbf{P}_\pi \mathbf{D}_v^{-1/2} \mathbf{C} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{C}^\top \mathbf{D}_v^{-1/2} \mathbf{P}_\pi^\top \\ &= \mathbf{P}_\pi \mathbf{L} \mathbf{P}_\pi^\top. \end{aligned} \tag{16}$$

Since $\mathbf{P}_\pi$ is orthogonal, $\mathbf{L}'$ is similar to $\mathbf{L}$, and thus

$$\mathrm{spec}(\mathbf{L}') = \mathrm{spec}(\mathbf{L}). \tag{17}$$

Therefore, there exist two distinct labeled hypergraphs $\mathcal{H} \neq \mathcal{H}'$ with identical spectra:

$$\Psi(\mathcal{H}) = \mathrm{spec}(\mathbf{L}) = \mathrm{spec}(\mathbf{L}') = \Psi(\mathcal{H}'). \tag{18}$$

Consequently, Ψ is a many-to-one mapping (non-injective), and its inverse reconstruction from spectral representations to labeled hypergraphs is not uniquely defined.

Hence, unique hypergraph reconstruction from spectral representations is ill-posed under standard incidence-based formulations. □

C.2 Restatement and Proof of Theorem 1

Theorem 1 (Phase enhancement strictly generalizes unsigned incidence)**.** *Let $\mathbf{C}^{\mathrm{phase}}_{ie} = \mathbf{C}_{ie} \exp(\mathrm{i}\phi_{ie})$ and define the induced Hermitian operator*

$$\mathbf{H}(\mathbf{C}^{\mathrm{phase}}) = \begin{pmatrix} \mathbf{0} & \mathbf{C}^{\mathrm{phase}} \\ (\mathbf{C}^{\mathrm{phase}})^\dagger & \mathbf{0} \end{pmatrix}.$$

If $\phi_{ie} = 0$ for all (i, e), then $\mathbf{H}(\mathbf{C}^{\mathrm{phase}})$ reduces to the unsigned real-valued case. Moreover, if there exist two distinct nodes $i \neq j$ such that $\sum_{e=1}^{m} \frac{\mathbf{C}_{ie}\mathbf{C}_{je} w(e)}{\delta(e)} \sin(\phi_{ie} - \phi_{je}) \neq 0$, then the interaction term $\left[\mathbf{C}^{\mathrm{phase}} \mathbf{W} \mathbf{D}_e^{-1} (\mathbf{C}^{\mathrm{phase}})^\dagger\right]_{ij}$ is non-real.

Proof. Let $\mathcal{L}_{\mathcal{H}}$ and $\mathcal{L}^{\mathrm{phase}}_{\mathcal{H}}$ denote the corresponding hypergraph Laplacians induced by $\mathbf{C}$ and $\mathbf{C}^{\mathrm{phase}}$, respectively.

When $\phi_{ie} = 0$ for all (i, e), we have $\mathbf{C}^{\mathrm{phase}} = \mathbf{C}$, and thus the phase-enhanced construction reduces to the unsigned case. This implies that any function realizable by HGConv can also be represented by $\mathrm{HGConv}^{\mathrm{phase}}$, and hence

$$\mathcal{F}_{\mathrm{HGConv}} \subseteq \mathcal{F}_{\mathrm{HGConv}^{\mathrm{phase}}}. \tag{19}$$

For general phase assignments, phase modulation can introduce additional complex-valued interaction terms in the Laplacian construction, yielding:

$$\mathcal{L}_{\mathcal{H}}^{\text{phase}} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{C}^{\text{phase}}\mathbf{W}\mathbf{D}_e^{-1}(\mathbf{C}^{\text{phase}})^{\dagger}\mathbf{D}_v^{-1/2}, \tag{20}$$

which can induce phase-dependent interference structures absent in the real-valued case.

In particular, if there exist two distinct nodes $i \neq j$ such that $\sum_{e=1}^{m} \frac{\mathbf{C}_{ie}\mathbf{C}_{je}w(e)}{\delta(e)} \sin(\phi_{ie} - \phi_{je}) \neq 0$, then the interaction term $\left[\mathbf{C}^{\text{phase}}\mathbf{W}\mathbf{D}_e^{-1}(\mathbf{C}^{\text{phase}})^{\dagger}\right]_{ij}$ is non-real, since $\operatorname{Im}\left(\left[\mathbf{C}^{\text{phase}}\mathbf{W}\mathbf{D}_e^{-1}(\mathbf{C}^{\text{phase}})^{\dagger}\right]_{ij}\right) = \sum_{e=1}^{m} \frac{\mathbf{C}_{ie}\mathbf{C}_{je}w(e)}{\delta(e)} \sin(\phi_{ie}-\phi_{je}) \neq 0$. The normalized Laplacian further satisfies

$$\operatorname{Im}\left([\mathcal{L}_{\mathcal{H}}^{\text{phase}}]_{ij}\right) = -\frac{1}{\sqrt{d(v_i)d(v_j)}} \sum_{e=1}^{m} \frac{\mathbf{C}_{ie}\mathbf{C}_{je}w(e)}{\delta(e)} \sin(\phi_{ie} - \phi_{je}) \neq 0. \tag{21}$$

Hence, the normalized phase-enhanced Laplacian contains a non-real interaction term. There exists an operator induced by $\mathrm{HGConv}^{\text{phase}}$ that cannot be expressed by HGConv, implying

$$\mathcal{F}_{\mathrm{HGConv}} \subsetneq \mathcal{F}_{\mathrm{HGConv}^{\text{phase}}}. \tag{22}$$

Furthermore, this increased expressive capacity benefits the parameterization of the time-conditioned vector field:

$$\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) = f_\psi\big(\mathrm{HGConv}^{\text{phase}}(\mathbf{Z}_t), \mathbf{e}_t\big). \tag{23}$$

□

C.3 Restatement and Proof of Theorem 2

Theorem 2 (Well-posed spectral flow under flow matching)**.** *Assume the learned vector field $\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$ is continuous in t, locally Lipschitz in $\mathbf{Z}$, and has at most linear growth on $t \in [0, 1]$. Then the ODE*

$$\frac{d\mathbf{Z}_t}{dt} = \mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$$

admits a unique solution on $[0, 1]$. Furthermore, for the noisy interpolation $\mathbf{Z}_t = (1 - t)\mathbf{Z}_0 + t\mathbf{Z}_1 + \sigma\sqrt{t}\,\boldsymbol{\varepsilon}$, we define the flow-matching regression target as

$$\mathbf{v}^\star(\mathbf{Z}_t, t) = \frac{\mathbf{Z}_1 - \mathbf{Z}_t}{1 - t}, \quad t \in [0, 1).$$

The regression target provides a reference for approximating the vector field at noisy latent states.

Proof. By the Picard–Lindelof theorem, the continuity of $\mathbf{v}_\psi$ in t and its local Lipschitz continuity in $\mathbf{Z}$ guarantee the existence and uniqueness of a local solution for any initial condition $\mathbf{Z}(0) = \mathbf{Z}_{\text{init}}$.

Moreover, since $\mathbf{v}_\psi$ satisfies the linear growth condition, Gronwall's inequality ensures that the solution remains bounded on $[0, 1]$. By the continuation theorem for ordinary differential equations, the local solution extends to the entire interval $[0, 1]$. Consequently, the ODE admits a unique solution throughout $[0, 1]$. Let $\mathbf{Z}_0 \sim p_0$ and $\mathbf{Z}_1 \sim p_1$. Consider the noisy interpolation path:

$$\mathbf{Z}_t = (1 - t)\mathbf{Z}_0 + t\mathbf{Z}_1 + \sigma\sqrt{t}\,\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad t \in [0, 1]. \tag{24}$$

For the noisy latent state $\mathbf{Z}_t$, the regression target is defined as

$$\mathbf{v}^\star(\mathbf{Z}_t, t) = \frac{\mathbf{Z}_1 - \mathbf{Z}_t}{1 - t}, \quad t \in [0, 1). \tag{25}$$

We parameterize the velocity field using a time-conditioned hypergraph neural network:

$$\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) = f_\psi\big(\mathrm{HGConv}(\mathbf{Z}_t, \mathbf{E}, w), \mathbf{e}_t\big). \tag{26}$$

The model is trained via the flow matching objective:

$$\mathcal{L}_{\text{spec-flow}}(\psi) = \mathbb{E}_{t,\mathbf{Z}_t}\left[\|\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) - \mathbf{v}^\star(\mathbf{Z}_t, t)\|_2^2\right]. \tag{27}$$

The objective is a squared loss regression problem. For any fixed $(\mathbf{Z}_t, t)$, the optimal predictor minimizes:

$$\mathbb{E}\left[\|\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) - \mathbf{v}^\star(\mathbf{Z}_t, t)\|_2^2 \mid \mathbf{Z}_t, t\right]. \tag{28}$$

Define $\mathbf{g} := \mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H})$, and consider the conditional objective:

$$\mathcal{J}(\mathbf{g}) = \mathbb{E}\left[\|\mathbf{g} - \mathbf{v}^\star(\mathbf{Z}_t, t)\|_2^2 \mid \mathbf{Z}_t, t\right]. \tag{29}$$

Expanding the squared norm:

$$\mathcal{J}(\mathbf{g}) = \|\mathbf{g}\|_2^2 - 2\,\mathbf{g}^\top \mathbb{E}\left[\mathbf{v}^\star \mid \mathbf{Z}_t, t\right] + \mathbb{E}\left[\|\mathbf{v}^\star\|_2^2 \mid \mathbf{Z}_t, t\right]. \tag{30}$$

Taking derivative with respect to $\mathbf{g}$:

$$\nabla_{\mathbf{g}}\mathcal{J}(\mathbf{g}) = 2\mathbf{g} - 2\mathbb{E}\left[\mathbf{v}^\star \mid \mathbf{Z}_t, t\right]. \tag{31}$$

Since $\mathcal{J}(\mathbf{g})$ is strictly convex and differentiable, its unique optimal solution is obtained by setting the gradient to zero:

$$\mathbf{g} = \mathbb{E}\left[\mathbf{v}^\star(\mathbf{Z}_t, t) \mid \mathbf{Z}_t, t\right]. \tag{32}$$

Thus, at the population optimum,

$$\mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}) = \mathbb{E}\left[\mathbf{v}^\star(\mathbf{Z}_t, t) \mid \mathbf{Z}_t, t\right]. \tag{33}$$

By the existence and uniqueness result established above, the learned vector field defines the drift ODE underlying the stochastic inference:

$$\frac{d\mathbf{Z}_t}{dt} = \mathbf{v}_\psi(\mathbf{Z}_t, t; \mathcal{H}). \tag{34}$$

Under the stated regularity assumptions, this ODE admits a unique solution on $[0, 1]$.

Finally, since $\mathbf{v}_\psi$ is parameterized via hypergraph convolution, the resulting vector field encodes high-order structural dependencies, completing the proof. □

D Training Algorithm and Time Complexity Analysis of HyperGen

As shown in Algorithms 1 and 2, we analyze the computational complexity of the two main training procedures in HyperGen.

Algorithm 1 Training Algorithm of Conditioned Hypergraph Spectral Vector Field in HyperGen

Require: Hypergraph incidence matrix $\mathbf{C}$, number of epochs E, noise scale σ, flow steps N
Ensure: Learned hypergraph-conditioned vector field $\mathbf{v}_\psi$

Apply Peierls phase to $\mathbf{C}$ to obtain complex incidence matrix $\mathbf{C}^{\text{phase}}$
Construct block-Hermitian matrix $\mathbf{H}$
Compute phase-aware spectral embedding

$$\mathbf{Z}_1 \leftarrow \text{SpectralEncoder}(\mathbf{H})$$

Extract hyperedge index $\mathbf{E}$ and weights $\mathbf{w}$ from $\mathbf{C}^{\text{phase}}$
Encode phase features $\mathbf{C}_{\text{feat}} = \{\Re(\mathbf{C}^{\text{phase}}), \Im(\mathbf{C}^{\text{phase}})\}$
Sample Gaussian latent prior

$$\mathbf{Z}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$$

for $epoch = 1$ **to** E **do**
    Sample time $t \sim \mathcal{U}(0, 1)$
    Sample noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Construct noisy interpolation:

$$\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t\mathbf{Z}_1 + \sigma\sqrt{t}\,\boldsymbol{\varepsilon}$$

    Compute flow-matching regression target:

$$\mathbf{v}^\star = \frac{\mathbf{Z}_1 - \mathbf{Z}_t}{1-t}$$

    Predict velocity via hypergraph-conditioned vector field:

$$\hat{\mathbf{v}} \leftarrow \mathbf{v}_\psi(\mathbf{Z}_t, t \mid \mathbf{E}, \mathbf{w}, \mathbf{C}_{\text{feat}})$$

    Compute flow matching loss:

$$\mathcal{L}_{\text{FM}} = \|\hat{\mathbf{v}} - \mathbf{v}^\star\|_2^2$$

    Update ψ by gradient descent
end for

The computational complexity of Algorithm 1 is dominated by the spectral embedding step. Let n be the number of nodes, m the number of hyperedges, and d the embedding dimension. The main contributions to the complexity are: applying the Peierls phase to the hypergraph incidence matrix $\mathbf{C}$ with $O(|\mathbf{C}|)$ cost, constructing the block-Hermitian matrix $\mathbf{H}$ with $O((n+m)^2)$, and computing the phase-aware spectral embedding via eigendecomposition with $O((n+m)^3)$ cost. Each training epoch (for a total of E epochs) involves sampling, vector field prediction, and gradient updates with per-epoch complexity $O((n+m)d^2 + P)$, where P is the number of learnable parameters.

Overall, Algorithm 1 has time complexity:

$$O\big((n+m)^3 + E \cdot ((n+m)d^2 + P)\big). \tag{35}$$

The computational complexity of Algorithm 2, which trains the hyperedge amplitude decoder, is much lighter. Let k be the average size of a hyperedge. Precomputing ground-truth hyperedge amplitudes costs $O(mk^2)$, while each training epoch (for E epochs) involves extracting hyperedge embeddings, MLP forward passes, and gradient updates with per-epoch cost $O(md^2 + P')$, where P' is the number of parameters in the amplitude decoder. Hence, the total complexity of Algorithm 2 is:

$$O\big(mk^2 + E \cdot (md^2 + P')\big). \tag{36}$$

In summary, Algorithm 1 is dominated by the spectral embedding step, whose cubic complexity in $(n+m)$ arises from eigendecomposition of the block-Hermitian operator. This leads to rapidly increasing computational cost with hypergraph size, making the framework less practical for large-scale settings and revealing a clear scalability limitation.

Algorithm 2 Training Algorithm of Hyperedge Amplitude Decoder

Require: Weighted adjacency matrix $\mathbf{A}$ of the projected graph, hyperedge set $\{\mathcal{V}_e\}_{e=1}^{m}$, spectral latent representations $\mathbf{Z}_1 \in \mathbb{R}^{(n+m)\times d}$, amplitude decoder $f_{\text{amp}}(\cdot)$, number of training epochs E

Ensure: Trained hyperedge amplitude decoder f_{amp}

Precompute ground-truth hyperedge amplitudes:

for $e = 1$ **to** m **do**

Compute hyperedge weight:

$$a_e = \frac{1}{|\mathcal{V}_e|^2} \sum_{i,j \in \mathcal{V}_e} \mathbf{A}_{ij}$$

end for

Linearly normalize amplitudes:

$$\tilde{a}_e = \frac{a_e - \min_e a_e}{\max_e a_e - \min_e a_e + \epsilon}, \quad \forall e$$

for $epoch = 1$ **to** E **do**

Extract hyperedge latent embeddings:

$$\mathbf{z}_e = \mathbf{Z}_1[n+e], \quad e = 1, \ldots, m$$

Predict normalized amplitudes:

$$\hat{a}_e = \sigma\big(f_{\text{amp}}(\mathbf{z}_e)\big), \quad \hat{a}_e \in (0,1)$$

Compute amplitude supervision loss:

$$\mathcal{L}_{\text{amp}} = \sum_{e=1}^{m} \big\|\hat{a}_e - \tilde{a}_e\big\|_2^2$$

Update f_{amp} by gradient descent

end for

E Details of Experimental Settings

E.1 Datasets

We evaluate the approaches on five real-world datasets from three distinct domains, including contact networks (contact-high-school and contact-primary-school), email networks (email-Eu and email-Enron), and a drug-related dataset (NDC-classes) [40]. All numerical results are reported to two decimal places, except those in the Statistical Properties of Generated Hypergraphs section, which are reported to three decimal places.

Each dataset is conceptualized as a universe consisting of three core components—nverts, simplices, and times—corresponding to the number of nodes, hyperedges (i.e., node sets representing multi-way interactions), and hyperedge occurrence timestamps, respectively, and we additionally report the node–hyperedge incidence statistics for each dataset. These components collectively define the corresponding hypergraph, with its statistics summarized in Table 5.

Table 5: Dataset statistics.

Dataset	Nodes	Node–hyperedges	Timestamped simplices	Unique simplices
contact-high-school	327	18,192	172,035	7,937
contact-primary-school	242	30,729	106,879	12,799
email-Enron	143	4,550	10,883	1,542
email-Eu	998	85,737	234,760	25,791
NDC-classes	1,161	6,443	49,724	1,222

E.2 Evaluation metrics

We evaluate the generated hypergraphs using the following metrics:

Structural / Incidence Metrics

- `Matched`: The number of correctly recovered node–hyperedge incidences between the generated hypergraph and the ground-truth hypergraph: Matched $= |\mathcal{I}_{\text{gen}} \cap \mathcal{I}_{\text{gt}}|$, where $\mathcal{I}_{\text{gen}}$ and $\mathcal{I}_{\text{gt}}$ denote the incidence sets of the generated and ground-truth hypergraphs, respectively.
- `Missing`: The number of ground-truth incidences that are not recovered: Missing $= |\mathcal{I}_{\text{gt}} \setminus \mathcal{I}_{\text{gen}}|$.
- `Extra`: The number of node–hyperedge relationships absent in the ground-truth hypergraph but present in the generated hypergraph: Extra $= |\mathcal{I}_{\text{gen}} \setminus \mathcal{I}_{\text{gt}}|$.
- `Recall`: The proportion of correctly recovered node–hyperedge relationships among all ground-truth relationships. It is computed based on the matched and missing incidences, given by: Recall $= \frac{\text{Matched}}{\text{Matched+Missing}}$.
- `Precision`: The proportion of correctly recovered node–hyperedge relationships among all relationships generated by the model. It is computed based on the matched and extra incidences, given by: Precision $= \frac{\text{Matched}}{\text{Matched+Extra}}$.
- `F1-score`: This metric measures the balance between precision and recall in evaluating node–hyperedge reconstruction quality, given by: F1 $= \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision+Recall}}$.

Spectral & Geometry Metrics

- `Eig-L2`: This metric measures the ℓ_2 distance between the leading eigenvalues of the adjacency matrices of the original and generated hypergraphs, capturing the deviation in global spectral structure: Eig-L2 $= \left\|\lambda^{\text{gt}} - \lambda^{\text{gen}}\right\|_2$.
- `Rel-Spec`: The relative spectral discrepancy by normalizing Eig-L2 with respect to the magnitude of the original spectrum: Rel-Spec $= \frac{\left\|\lambda^{\text{gt}} - \lambda^{\text{gen}}\right\|_2}{\left\|\lambda^{\text{gt}}\right\|_2 + \epsilon}$.
- `TopK-Spec`: The spectral discrepancy over the top-k eigenvalues, reflecting the difference in dominant spectral components: TopK-Spec $= \left\|\lambda^{\text{gt}}_{1:k} - \lambda^{\text{gen}}_{1:k}\right\|_2$.
- `Wasserstein`: The distributional distance between the eigenvalue spectra of the original and generated hypergraphs: Wasserstein $= \frac{1}{k}\sum_{i=1}^{k}\left|\lambda^{\text{gt}}_{(i)} - \lambda^{\text{gen}}_{(i)}\right|$.
- `Frobenius`: The overall structural deviation between adjacency matrices induced by incidence matrices: Frobenius $= \left\|\mathbf{C}^{\text{gt}}(\mathbf{C}^{\text{gt}})^\top - \mathbf{C}^{\text{gen}}(\mathbf{C}^{\text{gen}})^\top\right\|_F$.
- `SCC`: The misalignment between the principal spectral subspaces of the original and generated hypergraphs, defined based on the Frobenius norm of the projection matrix: SCC $= \sqrt{\max\left(2k - 2\left\|(U^{\text{gt}})^\top U^{\text{gen}}\right\|_F^2, 0\right)}$.

E.3 Implementation Details

Experimental Setup All experiments were conducted on a Linux system running on a machine equipped with an Intel(R) Xeon(R) Platinum 8468V CPU, 1 TB of memory, an NVIDIA H100 PCIe with 80 GB VRAM, and 7 TB SSD. The implementation is carried out in Python. We first construct a spectral representation of the hypergraph using fixed truncation hyperparameters, where the numbers of low-frequency and high-frequency components are set to $k_{low} = 16$ and $k_{high} = 16$, respectively. This yields a spectral embedding of dimension $3(k_{low} + k_{high})$, resulting in a 96-dimensional latent space that serves as the initial representation for the subsequent flow matching model and is used consistently throughout the generative process. In our method, the vector field network adopts a two-layer Hypergraph Convolution architecture, where the first layer maps the input features to a 128-dimensional hidden space and the second layer projects them into the latent dimension. In addition, we incorporate a temporal embedding module implemented as a two-layer MLP ($1 \to 64 \to 64$), as well as global hypergraph conditioning signals to modulate the dynamics of the vector

field. And HyperGen is trained using the Adam optimizer with a learning rate of 1×10^{-3} for 400 epochs. We perform threshold-based denoising on the generated probabilistic incidence matrix $\mathbf{C}_{\text{gen}}^{\text{prob}}$ to suppress low-confidence connections. Concretely, a threshold of 0.3 is applied, and entries below this value are set to zero, resulting in the denoised incidence matrix $\mathbf{C}_{\text{denoise}}$. For spectral evaluation, we measure the similarity between the generated and ground-truth hypergraphs in the spectral domain. The Frobenius metric is computed on the full adjacency matrices, while all other spectral metrics are evaluated using the top-k eigenvalues of the node-level adjacency matrix $\mathbf{S} = \mathbf{C}\mathbf{C}^\top$ for scalability. We set $k = 20$ in all experiments. For all baseline methods and the proposed HyperGen, we repeat each experiment five times with different random seeds to mitigate stochasticity and ensure fair comparison, reporting the averaged results over five runs.

Numerical Solver and Training Dynamics During training, we adopt a Flow Matching objective. Specifically, intermediate states are constructed via a stochastic interpolation path: $\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t\mathbf{Z}_1 + \sigma\sqrt{t}\,\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\sigma\sqrt{t}$ denotes the noise standard deviation and $\sigma = 0.2$. The regression target vector field is defined in closed form as $\mathbf{v}^*(\mathbf{Z}_t, t) = \frac{\mathbf{Z}_1 - \mathbf{Z}_t}{1-t}, t \in [0, 1)$. This target supervises the learned vector field via a mean squared error objective. Thus, HyperGen adopts conditional stochastic Flow Matching during training, with noise injected into the interpolation trajectory.

At inference time, we solve the learned stochastic dynamics using the Euler–Maruyama scheme with 20 integration steps and a fixed step size of $\Delta t = 0.05$. At each step, the latent state is updated as $\mathbf{Z}_{k+1} = \mathbf{Z}_k + \Delta t\, \mathbf{v}_\psi(\mathbf{Z}_k, t_k; \mathcal{H}) + 0.2\sqrt{\Delta t}\,\boldsymbol{\xi}_k, \boldsymbol{\xi}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here, $\boldsymbol{\xi}_k$ is independently sampled from a standard Gaussian distribution at each step. This update combines the learned drift with stochastic diffusion to generate latent representations for hypergraph decoding.

F Supplementary Experiments

In this section, we provide additional experimental results and analyses that were not included in the main text, to further complement and validate our findings.

F.1 Detailed results

Here, we report the results of HyperGen in terms of Recall, Precision, and F1, along with their standard deviations, computed over five independent runs with different random seeds.

Table 6: Detailed structural evaluation of HyperGen on temporal higher-order interaction networks.

Baseline	Dataset	Recall	Precision	F1
HyperGen	contact-high-school	1.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
	contact-primary-school	1.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00
	email-Enron	1.00 ± 0.00	0.02 ± 0.00	0.04 ± 0.00
	email-Eu	1.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00
	NDC-classes	1.00 ± 0.00	0.01 ± 0.00	0.01 ± 0.00

F.2 Structural and Spectral Evaluation

We further provide additional comparisons for the baseline methods HyperLap [14] and HyperDK00 [31], which are not discussed in the main text, in terms of structural reconstruction and spectral–geometric consistency. The results are reported in Tables 7 and 8.

From Table 7, it can be observed that both HyperLap and HyperDK00 struggle to accurately recover node–hyperedge relationships. In particular, HyperLap consistently exhibits low recall and F1 scores across all datasets, indicating its limited ability to capture node–hyperedge incidence structures. Although HyperDK00 achieves relatively higher recall on certain datasets (e.g., contact-high-school), this comes at the cost of substantially more extra relationships, resulting in very low precision and F1, suggesting that it tends to over-generate node–hyperedge connections without preserving true incidence patterns. From Table 8, both methods show significant discrepancies in spectral and

Table 7: Structural performance comparison on temporal higher-order interaction networks.

Baseline	Dataset	Matched	Missing	Extra	Recall	Precision	F1
HyperLap	contact-high-school	920	17272	17272	0.05	0.05	0.05
	contact-primary-school	1045	29684	29723	0.03	0.03	0.03
	email-Enron	108	4442	4460	0.02	0.02	0.02
	email-Eu	1172	84565	87609	0.01	0.01	0.01
	NDC-classes	8	6435	6460	0.00	0.00	0.00
HyperDK00	contact-high-school	6352	11840	268500	0.35	0.02	0.04
	contact-primary-school	2948	27781	329566	0.10	0.01	0.02
	email-Enron	461	4089	23316	0.10	0.02	0.03
	email-Eu	3673	82064	483456	0.04	0.01	0.01
	NDC-classes	53	6390	150171	0.01	0.00	0.00

Table 8: Spectral and geometric comparison on temporal higher-order interaction networks.

Baseline	Dataset	Rel-Spec	TopK-Spec	Wasserstein	Frobenius	SCC
HyperLap	contact-high-school	0.12	85.15	16.71	781.64	5.96
	contact-primary-school	26.59	1k	252.52	2k	5.95
	email-Enron	0.21	122.22	17.44	596.13	5.81
	email-Eu	0.25	1k	249.53	5k	6.17
	NDC-classes	0.63	873.11	75.27	1k	6.26
HyperDK00	contact-high-school	6.57	4k	880.69	14k	6.08
	contact-primary-school	98.28	4k	871.99	11k	5.92
	email-Enron	1.03	590.22	117.29	1738.02	5.69
	email-Eu	0.44	2k	337.98	13k	6.23
	NDC-classes	0.60	837.86	143.05	4k	6.24

geometric properties. HyperLap exhibits relatively stable spectral behavior on larger graphs, but degrades noticeably on smaller datasets. HyperDK00, on the other hand, exhibits consistently large spectral errors across all datasets, as reflected in its high Wasserstein distance and Frobenius norm. Moreover, the relatively high SCC values indicate poor alignment between the principal spectral subspaces of generated and ground-truth hypergraphs.

Overall, the results show that HyperLap [14] and HyperDK00 [31] either fail to accurately reconstruct node–hyperedge incidence relationships or suffer from severe spectral distortions, especially in large-scale and complex hypergraphs.

F.3 Statistical Properties of Generated Hypergraphs

To ensure a fair and comprehensive evaluation, we additionally adopt several statistics commonly used in prior work, including HyperPLR [17]. Specifically, density, average hyperedge size (avg. size), and average node degree (avg. degree) are used to characterize structural properties at the hypergraph level. The G coefficient and G modularity evaluate structural patterns on the projected graph, while B modularity measures structural organization in the bipartite representation. These metrics complement the main structural and spectral evaluations by providing coarse-grained statistical comparisons. In general, smaller discrepancies between the generated and ground-truth hypergraphs indicate higher generation fidelity. All reported metric values are rounded to three decimal places.

We compare representative baselines against the denoised version generated by HyperGen, and summarize the results in Table 9. Relative to non-spectral hypergraph generation methods, HyperGen consistently exhibits closer structural and spectral consistency with the ground-truth hypergraphs across hypergraph-level, projection-level, and bipartite-level evaluation metrics. This observation reveals the critical role of spectral information in recovering the structural characteristics of real-world hypergraphs and suggests that spectral-based hypergraph generation can achieve higher fidelity.

Among the non-spectral baselines, we observe that HyperLap, HyperLap+, and THERA exhibit competitive performance at the hypergraph structural level. HyperLap and HyperLap+ are able to

Table 9: Results of hypergraph generation in terms of graph metrics on five real-world hypergraph datasets. The table compares multiple methods, including the original real-world hypergraph (Real).

	Methods							
Metric	Real	HyperDK00	HyperDK11	HyperLap	HyperLap+	THERA	HyperPLR	HyperGen
				contact-high-school				
density	23.908	272.161	142.573	23.938	**23.908**	**23.908**	24.055	**23.908**
avg. size	2.327	3.089	2.178	**2.327**	**2.327**	2.609	2.703	**2.327**
avg. degree	55.633	840.796	310.567	55.701	**55.633**	62.382	65.011	**55.633**
G coefficient	0.504	0.999	0.861	0.341	0.632	0.354	0.471	**0.504**
G modularity	0.582	0.000	0.016	0.394	0.742	0.405	0.556	**0.582**
B modularity	0.754	0.324	0.459	0.622	0.747	0.831	0.649	**0.754**
				contact-primary-school				
density	52.496	229.896	143.343	**52.496**	**52.496**	**52.496**	52.714	**52.496**
avg. size	2.419	3.129	2.285	**2.419**	**2.419**	2.767	2.648	**2.419**
avg. degree	126.979	719.429	327.516	**126.979**	**126.979**	145.252	139.593	**126.979**
G coefficient	0.526	1.000	0.924	0.496	0.655	0.446	0.491	**0.526**
G modularity	0.284	0.000	0.010	0.281	0.496	0.150	0.281	**0.284**
B modularity	0.635	0.319	0.437	0.620	0.640	0.758	0.600	**0.635**
				email-Enron				
density	10.573	52.045	39.596	10.232	10.203	10.203	10.723	**10.573**
avg. size	3.009	3.188	2.885	3.131	3.131	3.386	3.244	**3.009**
avg. degree	31.818	165.933	114.252	32.034	31.944	34.545	34.782	**31.818**
G coefficient	0.593	0.958	0.796	0.646	0.841	0.674	**0.593**	**0.593**
G modularity	0.352	0.007	0.040	0.275	0.582	0.166	0.351	**0.352**
B modularity	0.656	0.322	0.355	0.576	0.738	0.736	0.569	**0.656**
				email-Eu				
density	25.077	156.130	114.040	25.778	25.860	24.868	29.081	**25.077**
avg. size	3.426	3.125	2.887	3.621	3.621	4.696	3.135	**3.426**
avg. degree	85.909	487.965	329.223	93.338	93.633	116.792	91.155	**85.909**
G coefficient	0.492	0.723	0.788	0.623	0.785	0.837	0.446	**0.492**
G modularity	0.371	0.015	0.035	0.280	0.745	0.099	0.319	**0.371**
B modularity	0.675	0.327	0.353	0.571	0.751	0.793	0.493	**0.675**
				NDC-classes				
density	0.937	36.112	31.083	1.089	1.098	0.999	1.210	**0.937**
avg. size	5.922	3.583	3.594	6.166	6.166	6.282	5.501	**5.922**
avg. degree	5.550	129.392	111.711	6.718	6.767	6.277	6.655	**5.550**
G coefficient	0.611	0.311	0.754	0.672	0.750	0.793	0.791	**0.611**
G modularity	0.610	0.038	0.058	0.456	0.793	0.520	0.623	**0.610**
B modularity	0.741	0.303	0.296	0.575	0.836	0.892	0.733	**0.741**

generate hypergraphs whose density, average hyperedge size, and average node degree closely match those of the ground-truth hypergraphs. We attribute this to their ability to reproduce the same overlap patterns as the original hypergraphs. THERA's community-aware generative mechanism captures structural patterns, including multi-level transitivity, while preserving global structural statistics. Meanwhile, HyperPLR achieves strong performance on the graph projection and bipartite representation levels. The result aligns with the method's design, as HyperPLR explicitly projects hypergraphs onto weighted graphs and learns the hypergraph structure in the projected space, preserving pairwise and mesoscopic structural properties.

F.4 Spectral Visualization of Generated Hypergraphs

We perform a spectral visualization experiment to qualitatively investigate the spectral properties of the generated hypergraphs. Under identical experimental settings on the contact-primary-school dataset, both the original hypergraph and those generated by all models were represented as bipartite adjacency matrices, and their spectral properties were visualized and analyzed.

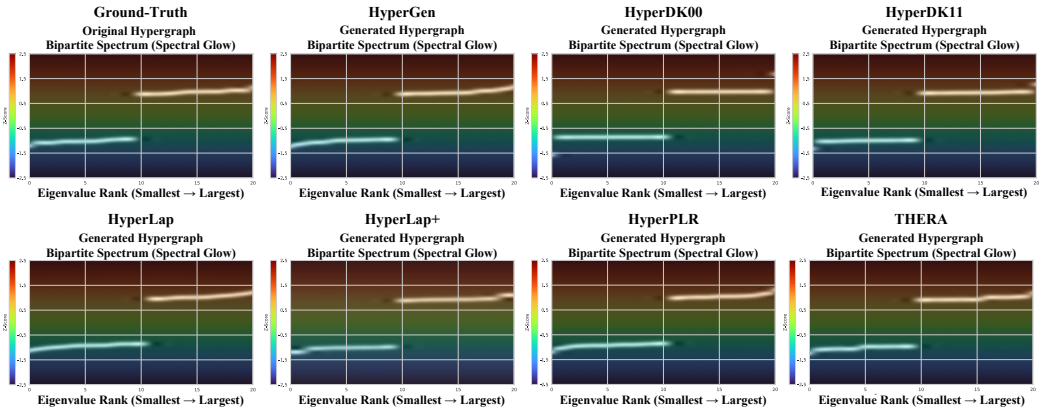

Figure 2: Spectral eigenvalue analysis on the contact-primary-school.

For bipartite adjacency matrices, the spectrum reflects high-order node–hyperedge interactions, and consistency with the spectral profile of the ground-truth indicates that high-order interactions are preserved. Eigenvalues appear in symmetric positive–negative pairs, corresponding to the same structural pattern but differing only in the relative sign (phase) between the node and hyperedge subspaces. The magnitude of an eigenvalue reflects the variation of its eigenvector across nodes and hyperedges: large magnitudes indicate rapid changes and emphasize prominent structural patterns across hyperedges (high-frequency modes), while small magnitudes correspond to smooth variations, capturing coherent patterns within hyperedges (low-frequency modes). By examining the distribution of eigenvalue magnitudes, we assess the dominant high-order interaction patterns captured by the generated hypergraphs. We extract the top-k ($k = 20$) eigenvalues $\{\lambda_i\}_{i=1}^{k}$ with the largest absolute magnitudes from the bipartite adjacency matrix $\mathbf{A}$ to characterize the dominant spectral components of node–hyperedge interactions. To facilitate visualization and comparison, each eigenvalue is standardized using the z-score, computed as follows: $\lambda_i^{(z)} = \frac{\lambda_i - \mu_\lambda}{\sigma_\lambda}, \mu_\lambda = \frac{1}{k}\sum_{i=1}^{k}\lambda_i, \sigma_\lambda = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(\lambda_i - \mu_\lambda)^2}$. As shown in Figure 2, the horizontal axis corresponds to the eigenvalue rank, ordered from smallest to largest, and is displayed on a continuous scale from 0 to 20 following horizontal interpolation. The vertical axis represents the standardized eigenvalues ($z - scores$), ranging from -2.5 to 2.5. To depict the overall spectral distribution, each eigenvalue is represented by a Gaussian curve along the vertical axis, and the curves are superimposed to form a gray "spectral glow," highlighting the dominant spectral components.

We clearly observed that the spectral line patterns and their smoothness in the HyperGen generated hypergraphs are highly consistent with those of the ground truth. This spectral consistency explains why HyperGen can both reproduce the eigenvalue magnitude distribution of the original hypergraph and preserve the underlying spectral characteristics associated with high-order node–hyperedge interactions. By contrast, the non-spectral generation methods, namely HyperDK00, HyperDK11, HyperLap+, and THERA, produce spectra with a substantial number of repeated eigenvalues, resulting in highly concentrated and nearly uniform spectral visualizations that manifest as near-linear or flat patterns. The observations suggest a tendency toward spectral degeneration in non-spectral hypergraph generation methods, potentially leading to the loss of structurally informative modes in the spectral domain. Although HyperLap and HyperPLR produce similarly smooth spectral curves, their eigenvalue distributions still deviate noticeably from the ground truth. Consequently, the hypergraphs reconstructed from overlapping and projection-based patterns fail to accurately preserve the dominant structural modes associated with high-order node–hyperedge interactions, suggesting a deviation of spectral information from the original hypergraph.

F.5 2D-layout Visualization of Latent Node Representations

We visualize and compare the distribution of node-level latent representations generated by HyperGen with that of the ground truth. We extracted node-level latent vectors from both the original and

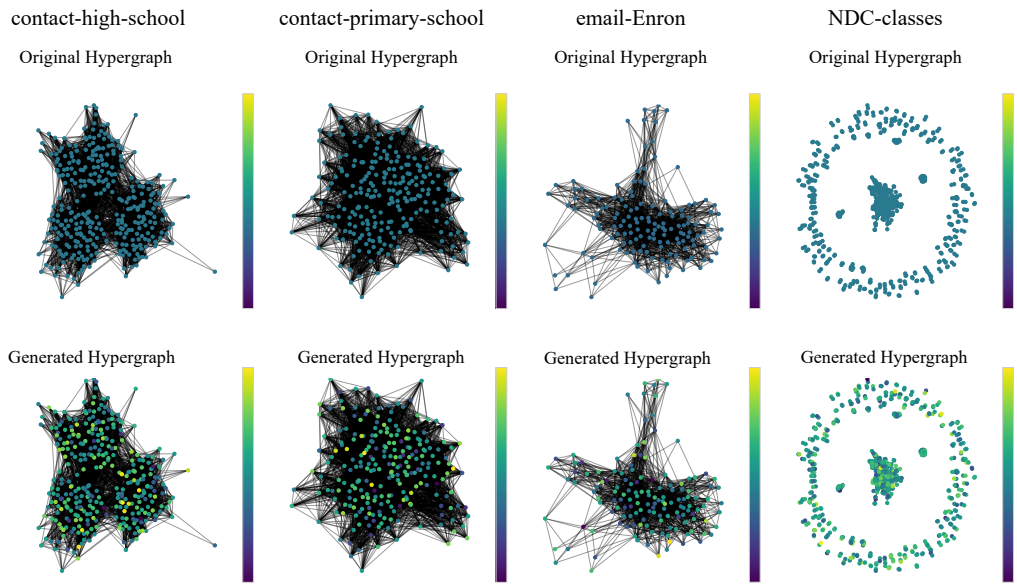

Figure 3: Visualization of projected node latent representations for the original and HyperGen-generated hypergraphs, illustrating both the consistency of relative node distributions and the semantic richness of the generated nodes.

generated hypergraphs and computed their ℓ_2 norms as node energy, reflecting the strength or prominence of each node in the latent space. Node colors encode these norms, with larger values mapped to brighter colors under a shared colormap.

To ensure a fair comparison, both the original and generated hypergraphs are visualized using an identical spring layout and a shared color range, eliminating potential visual bias. Node colors represent latent energy. By jointly considering node colors and positions, we can assess whether the generated latent representations preserve structural patterns and semantic roles of nodes, providing a visual evaluation of HyperGen's ability to maintain high-order structural and semantic information.

As shown in Figure 3, HyperGen is evaluated on four datasets: contact-high-school, contact-primary-school, email-Enron, and NDC-classes. In each case, the spatial distribution of node latent representations in the projected hypergraphs generated by HyperGen closely aligns with that of the original hypergraph. Nodes with high and low latent energies maintain stable relative positions within the graph structure, indicating that the model consistently preserves the relative placement of nodes in the latent space across different datasets.

Visualization of node latent representations further reveals that HyperGen not only captures the distribution of accurate projected graph structures but also models semantic relationships among nodes in the latent space.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We state our contributions in the abstract and at the end of the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section 6, we elaborate on the limitations of HyperGen.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The theoretical results are supported by clearly stated assumptions, and their formal proofs are provided in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Besides code, the paper provides detailed descriptions of the experimental setup, including model architecture, training procedure, and optimization settings, all of which are provided in Appendix E.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: As stated in the abstract, we provide a link to access the code. All resources required to reproduce the experiments, including training scripts, datasets, data processing pipelines, and detailed usage instructions, are provided in the README, thereby ensuring that readers can reliably reproduce the main experimental results under the settings reported in the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides all essential training details in Appendix E.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide error bars for Recall, Precision, and F1 in Appendix F.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides detailed information in Appendix E.3 about the computational resources used, including hardware configuration (CPU, GPU, memory), training settings, and optimization procedures, sufficient to understand the computational requirements of the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics and does not involve any ethical concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: The proposed method is a general-purpose framework for hypergraph generation, without being evaluated in specific downstream applications, and its potential societal implications are not explicitly analyzed in this work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release models or datasets with potential misuse risks and therefore does not require additional safeguards for responsible release.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper uses publicly available datasets released by their original authors. All datasets are properly cited, and their usage follows the corresponding licenses and terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The relevant documentation will be provided alongside the code.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.