PowerGrow: Feasible Co-Growth of Structures and Dynamics for Power Grid Synthesis

Xinyu He
University of Illinois at Urbana-Champaign
Champaign, USA
xhe34@illinois.edu

Chenhan Xiao
Arizona State University
Tempe, USA
cxiao20@asu.edu

Haoran Li
Massachusetts Institute of Technology
Cambridge, USA
haorandd@mit.edu

Ruizhong Qiu
University of Illinois at Urbana-Champaign
Champaign, USA
rq5@illinois.edu

Xu Zhe
Meta
Sunnyvale, USA
zhexu@meta.com

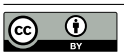
Yang Weng
Arizona State University
Tempe, USA
yweng2@asu.edu

Jingrui He
University of Illinois at Urbana-Champaign
Champaign, USA
jingrui@illinois.edu

Hanghang Tong
University of Illinois at Urbana-Champaign
Champaign, USA
htong@illinois.edu

Abstract

Modern power systems are becoming increasingly dynamic, with changing topologies and time-varying loads driven by renewable energy variability, electric vehicle adoption, and active grid reconfiguration. Despite these changes, publicly available test cases remain scarce, due to security concerns and the significant effort required to anonymize real systems. Such limitations call for generative tools that can jointly synthesize grid structure and nodal dynamics. However, modeling the joint distribution of network topology, branch attributes, bus properties, and dynamic load profiles remains a major challenge, while preserving physical feasibility and avoiding prohibitive computational costs. We present PowerGrow, a co-generative framework that significantly reduces computational overhead while maintaining operational validity. The core idea is dependence decomposition: the complex joint distribution is factorized into a chain of conditional distributions over feasible grid topologies, time-series bus loads, and other system attributes, leveraging their mutual dependencies. By constraining the generation process at each stage, we implement a hierarchical graph beta-diffusion process for structural synthesis, paired with a temporal autoencoder that embeds time-series data into a compact latent space, improving both training stability and sample fidelity. Experiments across benchmark settings show that PowerGrow not only outperforms prior diffusion models in fidelity and diversity but also achieves a 98.9% power flow convergence rate and improved N-1 contingency resilience. This demonstrates its ability to generate operationally valid and realistic power grid scenarios.



KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2258-5/2026/08
https://doi.org/10.1145/3770854.3780235

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**.

Keywords

Power Grid Synthesis; Graph Generation; Time Series Data Generation; Diffusion Model

ACM Reference Format:
Xinyu He, Chenhan Xiao, Haoran Li, Ruizhong Qiu, Xu Zhe, Yang Weng, Jingrui He, and Hanghang Tong. 2026. PowerGrow: Feasible Co-Growth of Structures and Dynamics for Power Grid Synthesis. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770854.3780235

1 Introduction

Modern power grids are undergoing rapid structural and temporal changes due to the integration of renewable energy, electric vehicles, and demand-side technologies [8, 39]. These trends introduce significant variability in both network topology and nodal power injections, requiring accurate models to simulate system-wide behavior under evolving conditions. However, publicly available test cases remain scarce due to data sensitivity and the substantial effort needed to anonymize real systems and construct realistic scenarios [3, 49]. As a result, synthetic datasets are increasingly used to support forecasting, planning, and resilience studies.

There is extensive existing work on producing synthetic grid structures and time-series dynamics, such as load profiles [7, 38]. For example, traditional load and topology synthesis methods often rely on statistical models [1, 2, 30], which fail to capture the complex spatialtemporal dependencies in modern power systems. While deep learning models such as Graph Neural Networks (GNNs) [41] and Long Short-Term Memory (LSTM) networks [10] have

demonstrated promise in learning from structured and sequential data, they typically disregard physical laws and operational feasibility constraints. As a result, outputs from these models often lack fidelity or violate grid constraints, limiting their applicability in real-world settings [30]. In general, existing approaches usually treat topology generation and load profile synthesis as two separate tasks [20, 33, 34, 46].

However, time-series operational data and graph structures exhibit strong interdependence due to inherent topology consistency [5, 9, 14, 29]. For example, in operational practice, power systems are continuously reconfigured both topologically and temporally to ensure reliability and efficiency [35]. Hence, it is important to align and evaluate these two types of generation together. Sequential pipelines that first generate a grid and then simulate loads have been proposed [28], but they fail to enforce physical consistency: generated loads may violate feasibility constraints under the assumed topology, and the topology itself may not reflect the influence of underlying demand patterns [26]. This fact leads to infeasible operating points, voltage violations, and artificial use cases that are highly unlikely to happen, which compromise planning, simulation, or control tasks.

Therefore, some cutting-edge work [45] introduces post-process efforts, e.g., conducting mixed-integer programming, to correctly combine generated graph structures, physical parameters, and load curves into a feasible and operable system. This optimization-based procedure is computationally expensive, which can not scale to large systems. In this paper, we explore the strong correlations between graphs and time series to exploit conditional dependence for reducing the generation space and generation time.

Specifically, we investigate diffusion models for power networks as they offer a compelling foundation: recent advances in graph diffusion models [17, 18, 27, 44] have demonstrated strong performance across various structured domains such as molecule design [15, 23], social networks, etc. However, as shown in Table 1, these methods suffer significant performance degradation when directly applied to complicated physical systems, such as power systems. First, unlike domains such as molecular generation where node and edge features are typically discrete (e.g., atom and bond types) [15], power systems exhibit a mix of tightly coupled discrete (e.g., bus types, topology connections), continuous (e.g., line impedance, power injections), and temporal data (e.g., nodal load profiles). Modeling such heterogeneous data in a unified way poses a high-dimensional, interdependent generation problem that overwhelms standard end-to-end diffusion architectures. Second, nodal load profiles in power systems span long temporal horizons and exhibit complex dependencies, including daily and weekly cycles [12], spatial correlations among neighboring buses [22], etc. A generative model should also accurately capture these dependencies. Third, there is a feasibility concern for the above multi-type data generation.

To manage this complexity, we propose a hierarchical Graph Beta Diffusion framework that employs a conditional generation strategy: discrete components such as bus types and structural connectivity are synthesized first, followed by continuous attributes like branch impedances and temporal attributes like load profiles. This coarse-to-fine factorization aligns with real-world grid formation and significantly improves generation fidelity and feasibility. Moreover, to efficiently capture temporal features, we pretrain a Long Short-Term Memory Autoencoder (LSTM-AE) to extract compact latent representations. These embeddings serve as dynamic node features that are integrated into the diffusion-based structural generation, enabling coherent modeling across both spatial and temporal dimensions. Then, the diffusion model and the decoder in LSTM-AE are fine-tuned to capture the interdependence and achieve alignment between the graph and the time series. Finally, to achieve high feasibility, we employ a domain-specific power flow simulator [6, 40] to prepare high-quality training data.

Overall, a three-level hierarchical graph beta diffusion framework is proposed to decompose the joint generation task into three conditional generation subtasks. We refer to this three-level diffusion framework as **PowerGrow**, whose pipeline is shown in Fig. 1. In summary, our main contributions are as follows:

- **Joint Structural–Temporal Synthesis**: We tackle the underexplored task of co-generating realistic grid topologies and nodal load profiles, explicitly modeling their mutual constraints to ensure operational feasibility without costly post-processing.
- **Hierarchical Dependence Decomposition**: We propose PowerGrow, a three-stage Graph Beta Diffusion framework that factorizes generation into structure, branch attributes, and temporal loads, mirroring real-world grid formation and improving scalability and fidelity.
- **Efficient Long-Horizon Load Modeling**: We integrate an LSTM-based temporal autoencoder to compress load profiles into compact node embeddings, reducing generation cost and preserving temporal coherence.
- **Strong Empirical Results**: On benchmark systems, PowerGrow achieves <0.01 MMD scores, 98.9% power flow convergence rate, and improved N-1 resilience, producing operationally valid grids that outperform prior generative models in both fidelity and diversity.

2 Related Work

Power grid topology and load synthesis. Generating synthetic power grid topologies and load profiles is becoming increasingly important for scalable analysis, system design, and planning [5, 9, 20, 33, 34, 46]. Early topology synthesis used VAEs to model grid structure [20], later methods extended to preserve structural motifs [9] and incorporate geographic constraints [5, 34]. In parallel, load generation has leveraged GANs to capture spatiotemporal patterns in historical data [14, 29, 46].

However, most existing approaches treat topology and load synthesis as decoupled tasks [20, 33, 34, 46], ignoring their physical interplay [5, 9, 14, 29]. This can lead to infeasible grids [26]. For example, generating a weakly connected topology followed by heavy industrial loads may violate voltage or thermal limits, while overlaying low-variability residential loads onto a high-redundancy mesh grid leads to overdesign and inefficiency. We bridge this gap by proposing a hierarchical co-generation framework that integrates multi-level Graph Beta Diffusion (GBD) with a temporal autoencoder. This joint modeling explicitly captures the structural-temporal interplay.

Graph diffusion generative models. Graph diffusion generative models have attracted growing interest recently owing to their strong sample fidelity and stable training. Early efforts such as EDP-GNN [27] and GDSS [18] model graph data in a continuous state space, requiring quantization during sampling to recover discrete structures. To better align with the inherently discrete nature of graphs, methods like DiGress [36] and DisCo [44] adopt discrete-state diffusion processes, where edges and node types evolve categorically during noise corruption. Several extensions explore more flexible diffusion paradigms: EDGE [4] diffuses the clean graphs to empty graphs, different from typical random graphs; GraphArm [21] introduces absorbing states through masked nodes and edges, framing the diffusion process in an autoregressive manner; Graph Beta Diffusion [24] applies a beta diffusion process that accommodates both discrete graph structures and continuous node attributes.

Although graph diffusion models are widely used in fields such as bioinformatics [11, 13, 16, 37] and materials science [19, 43, 47], applying them directly to power grid generation is nontrivial. Power grid data combines discrete topology with continuous node/edge attributes and temporal features. Generating synthetic grid graphs that maintain physical feasibility, discrete structure, and temporal coherence remains a complex and largely unexplored problem.

3 Preliminaries

Notations and problem formulation. We represent a power grid as a graph $\mathcal{G} = (X, A, E, D)$, where: $X \in \mathbb{R}^{N\times d_1}$ denotes the d_1-dimensional node features for N nodes (buses), including the node types (generator bus or load bus) and nodal constraint parameters, such as maximum generation limits, ramp rates, and unit costs. $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix specifying the grid topology. To align with the shape of the adjacency matrix, we formulate edge (branch) features as $E \in \mathbb{R}^{N\times N\times d_2}$, where $E[i, j]$ denotes the d_2-dimensional feature of the branch between bus i and j, including electrical parameters (e.g., resistance, reactance) and operational constraints (e.g., line power flow limits). Finally, we denote by $D \in \mathbb{R}^{N\times T\times d_3}$ the nodal-level time-series matrix containing power grid load profiles over T time steps.

This framework is flexible and can be extended to incorporate additional node- or branch-level features. For instance, X may further include voltage setpoints, while E can be augmented with parameters for advanced transmission components such as series compensation devices or flexible AC transmission systems (FACTS), which are commonly used to mitigate resonance and improve power flow controllability.

Let $\mathbf{v}$ denote the vector of physical and operational constraint violations defined over buses and branches (detailed constraints discussed in Appendix A), our objective is to develop a generative model to generate $\tilde{\mathcal{G}}$ satisfying two key criteria: (1) *fidelity*: the distribution of $\tilde{\mathcal{G}}$ should be close to that of real grids $\mathcal{G}$ and (2) *feasibility*: the topology and loads in $\tilde{\mathcal{G}}$ should have minimal violations in $\mathbf{v}$. In Section 5, we further quantify the feasibility by mapping $\mathbf{v}$ to a continuous feasibility score that reflects the severity of constraint violations.

Graph beta diffusion. Denoising diffusion models are composed of two processes: a forward diffusion process and a backward denoising process. In the forward process, the input data is gradually diffused into near-zero values or pure noise by applying noise sampled from a fixed noise distribution (e.g., Gaussian distribution, Beta distribution) step-by-step. Conversely, the reverse process learns to reconstruct the input data from noise using trainable neural networks. Compared to Gaussian noise, Beta-distributed noise better models the characteristics of graph data relevant to power systems: it naturally captures structural sparsity, long-tailed feature distributions, and bounded attribute ranges [24]. In graph beta diffusion [24], the forward multiplicative K-step beta diffusion process [48] for a plain graph is formulated as: $\forall k \in \{1, 2, \ldots, K\}$,

$$\begin{aligned} A_k &= A_{k-1} \odot Q_{A,k}, \quad X_k = X_{k-1} \odot Q_{X,k}, \\ Q_{A,k} &\sim \texttt{Beta}(\eta_A \alpha_k A_0, \eta_A(\alpha_{k-1} - \alpha_k)A_0), \\ Q_{X,k} &\sim \texttt{Beta}(\eta_X \alpha_k X_0, \eta_X(\alpha_{k-1} - \alpha_k)X_0), \end{aligned} \tag{1}$$

where Beta is beta distribution, η_A and η_X adjust the concentration of beta distributions, and α_k is the noise schedule at step k, $(A_0, X_0) = (A, X)$ are the input adjacency and feature matrices. Based on the beta diffusion process defined in Eq. (1), the marginal distribution for forward process is

$$\begin{aligned} q(A_k|A_0) &= \texttt{Beta}\left(\eta_A \alpha_k A_0, \eta_A(1 - \alpha_k A_0)\right), \\ q(X_k|X_0) &= \texttt{Beta}\left(\eta_X \alpha_k X_0, \eta_X(1 - \alpha_k X_0)\right). \end{aligned} \tag{2}$$

To equivalently construct the trajectory in the reverse order, the reverse multiplicative sampling process is formulated as

$$\begin{aligned} A_{k-1} &= A_k + \delta A_k, \quad X_{k-1} = X_k + \delta X_k \\ \delta A_k &= P_{A,k} \odot (1 - A_k), \quad \delta X_k = P_{X,k} \odot (1 - X_k), \\ P_{A,k} &\sim \texttt{Beta}(\eta_A(\alpha_{k-1} - \alpha_k)A_0, \eta_A(1 - \alpha_{k-1}A_0)), \\ P_{X,k} &\sim \texttt{Beta}(\eta_X(\alpha_{k-1} - \alpha_k)X_0, \eta_X(1 - \alpha_{k-1}X_0)). \end{aligned} \tag{3}$$

The reverse process, defined via ancestral sampling, is:

$$\begin{aligned} p_\alpha(A_{k-1}|A_k) &= \frac{1}{1 - A_k}\texttt{Beta}\left(\frac{A_{k-1} - A_k}{1 - A_k}\middle|\eta_A(\alpha_{k-1} - \alpha_k)\hat{A}_0 \right. \\ &\qquad \left., \eta_A(1 - \alpha_{k-1}\hat{A}_0)\right), \\ p_\alpha(X_{k-1}|X_k) &= \frac{1}{1 - X_k}\texttt{Beta}\left(\frac{X_{k-1} - X_k}{1 - X_k}\middle|\eta_X(\alpha_{k-1} - \alpha_k)\hat{X}_0 \right. \\ &\qquad \left., \eta_X(1 - \alpha_{k-1}\hat{X}_0)\right), \\ \left(\hat{A}_0, \hat{X}_0\right) &= \mathbb{E}[A_0, X_0 \mid A_k, X_k, k], \end{aligned} \tag{4}$$

where $\frac{1}{1-A_k}$ stands for element-wise inversion of $1 - A_k$. In practice, a neural network $f_\alpha()$ is trained to predict the expectation of $\mathcal{G}_0$ given $\mathcal{G}_k$, $\mathbb{E}[A_0, X_0 \mid A_k, X_k, k] \approx f_\alpha(A_k, X_k, k)$.

4 Method

In this section, we first tackle the efficiency problem for lone-span time sequence diffusion through time series modeling in Section 4.1. Section 4.2 then introduces the hierarchical beta diffusion model for decomposing complex joint generation problem into three simpler conditional generation subtasks. Section 4.3 summarizes the training paradigm and overall framework of our PowerGrow algorithm.

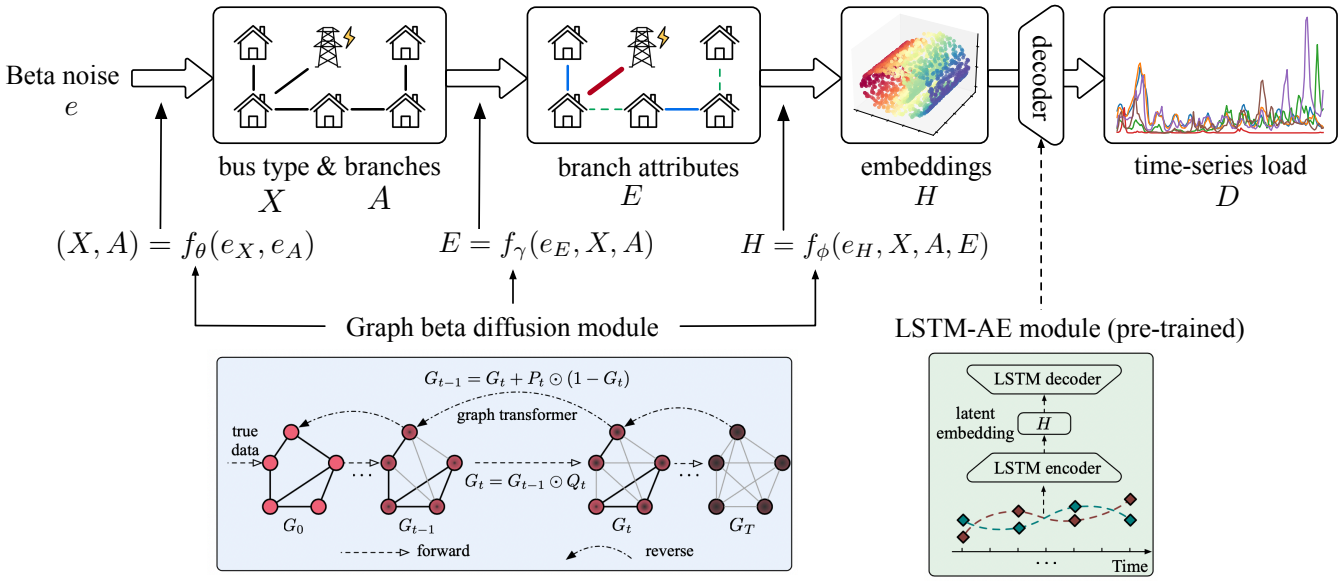


Figure 1: Visualization of the sampling process of the proposed PowerGrow framework. The framework is composed of: (1) *LSTM Autoencoder* module, and (2) *three graph transformers* (graph beta diffusion module). The training of the three graph transformers is done in parallel. The encoder in pretrained LSTM-AE is frozen for encoding time-series, but the decoder is finetuned along with the training of diffusion module. For Sampling, A and X, E, H, are sampled sequentially.

4.1 Time Series Modeling

Loads in power grids often exhibit clear periodic patterns which can be captured by sequential models. In this work, PowerGrow enables a co-generation of bus loads and power grids topology, which involves the 4 factorized distributions in the training of diffusion models. A direct adaptation of graph beta diffusion [24] to joint power grid synthesis in forward process is formulated as

$$\begin{aligned} q(A_k|A_0) &= \texttt{Beta}\left(\eta_A\alpha_k A_0, \eta_A(1-\alpha_k A_0)\right), \\ q(E_k|E_0) &= \texttt{Beta}\left(\eta_A\alpha_k E_0, \eta_A(1-\alpha_k E_0)\right), \\ q(X_k|X_0) &= \texttt{Beta}\left(\eta_X\alpha_k X_0, \eta_X(1-\alpha_k X_0)\right). \\ q(D_k|D_0) &= \texttt{Beta}\left(\eta_X\alpha_k D_0, \eta_X(1-\alpha_k D_0)\right). \end{aligned} \tag{5}$$

where we assume that A, E and X, D share the same concentration η_A and η_X, e.g., A and E should share the same sparsity. This forward diffusion process becomes increasingly expensive with the growth of T from tens to thousands, because the complexity of diffusing $\{D_k\}_{k=1}^T$ is $O(NTd_3)$. To facilitate efficient diffusion model learning and generation, we are inspired by the latent diffusion [32] to encode the power loads into a temporal embedding vector $H \in \mathbb{R}^{N\times d_4}$ via a sequential model, where $d_4 = 32$ in our experiments. Specifically, an LSTM-based autoencoder is applied:

$$H = g_{\text{enc}}(D), \quad \hat{D} = g_{\text{dec}}(H), \tag{6}$$

where g_{enc} and g_{dec} are two LSTMs. We pretrain this LSTM-AE via reconstruction loss for two motivations: (1) to avoid the cold-start problem and ensure stable training: in early stage of training, H does not hold meaningful information from D, which hinders the learning of diffusion model, and (2) avoid constructing extra large computational graphs from automatic differentiation engine (e.g., PyTorch) whose backpropagation is expensive. Therefore a three-stage pipeline is applied for training. First, LSTM-AE is trained in an unsupervised way with Mean Squared Error (MSE). Then diffusion model will take embeddings of dynamic features as inputs for generation, i.e., instead of diffusing $q(D_k|D_0)$, we will diffuse $q(H_k|H_0) = \texttt{Beta}\left(\eta_X\alpha_k H_0, \eta_X(1-\alpha_k H_0)\right)$. Lastly, diffusion model and LSTM-AE will be finetuned alternatively to better align embedding space of LSTM-AE with hidden space of diffusion model. In this way, we ensure a more stable training process, and leverage the interdependency between bus loads and other factors.

4.2 Hierarchical Beta Diffusion

Through ancestral sampling, a direct adaptation of reverse diffusion process involves a trainable network f_α for predicting

$$\left(\hat{A}_0, \hat{X}_0, \hat{E}_0, \hat{H}_0\right) = f_\alpha(A_k, X_k, E_k, H_k, k) \tag{7}$$

at diffusion step k. Previous graph diffusion models mainly focus on categorical features in molecule generation, or node-attributed graph. In this work, the co-generation of numerical edge features and time-series bus loads poses a new challenge for graph diffusion models. Our experiments (Table. 1) show that existing methods fail to properly learn the dependencies between the four factors. To tackle this challenge, we decompose the complex joint generation task into three conditional generation tasks, which aligns with the hierarchical causal dependency between power structure and bus types, edge features, and bus loads. We decompose the distribution

P_α that f_α is trying to learn as

$$\begin{aligned}
&P_\alpha\left(\hat{A}_0, \hat{X}_0, \hat{E}_0, \hat{H}_0 \middle| A_k, X_k, E_k, H_k, k\right) \\
&= P_\phi\left(\hat{H}_0 \middle| \hat{A}_0, \hat{E}_0, \hat{X}_0, A_k, X_k, E_k, H_k, k\right) \cdot \\
&\quad P_\gamma\left(\hat{E}_0 \middle| \hat{A}_0, \hat{X}_0, A_k, X_k, E_k, H_k, k\right) \cdot P_\theta\left(\hat{A}_0, \hat{X}_0 \middle| A_k, X_k, E_k, H_k, k\right) \\
&= P_\phi\left(\hat{H}_0 \middle| \hat{A}_0, \hat{E}_0, \hat{X}_0, H_k, k\right) \cdot P_\gamma\left(\hat{E}_0 \middle| \hat{A}_0, \hat{X}_0, E_k, H_k, k\right) \cdot \\
&\quad P_\theta\left(\hat{A}_0, \hat{X}_0 \middle| A_k, X_k, E_k, H_k, k\right). \qquad (8)
\end{aligned}$$

Here we assume that A_k and X_k contain no additional predictive information beyond $\hat{A}_0$ and $\hat{X}_0$ for estimating $\hat{E}_0$ and $\hat{H}_0$, as they represent noisy perturbations. A similar assumption applies to E_k for predicting $\hat{H}_0$. Furthermore, we assume that the generation of A, X is independent of E, H, and that E is independent of H, consistent with the underlying hierarchical causal structure: topology and node types are first sampled, followed by branch attributes, and finally load profiles, each conditioned only on prior stages. Based on the independency assumption, Eq. (8) is simplified to

$$\begin{aligned}
P_\alpha\left(\hat{A}_0, \hat{X}_0, \hat{E}_0, \hat{H}_0 \middle| A_k, X_k, E_k, H_k, k\right) &= P_\phi\left(\hat{H}_0 \middle| \hat{A}_0, \hat{E}_0, \hat{X}_0, H_k, k\right) \cdot \\
P_\gamma\left(\hat{E}_0 \middle| \hat{A}_0, \hat{X}_0, E_k, k\right) \cdot P_\theta\left(\hat{A}_0, \hat{X}_0 \middle| A_k, X_k, k\right). \qquad (9)
\end{aligned}$$

From Eq. (9), the joint generation task is naturally decomposed into three conditional generation subtasks: the generation of grid structure, generation of branch features conditioned on grid structure, and generation of bus loads conditioned on grid structure and branch features. Therefore, the trainable module in reverse diffusion process is formulated as

$$\begin{aligned}
\left(\hat{A}_0, \hat{X}_0\right) &= f_\theta(A_k, X_k, k), \\
\hat{E}_0 &= f_\gamma(A_0, X_0, E_k, k), \\
\hat{H}_0 &= f_\phi(A_0, X_0, E_0, H_k, k),
\end{aligned} \qquad (10)$$

where $f_\theta, f_\gamma, f_\phi$ are implemented as graph transformers [36]. This three-level decomposition allows us to train the three graph transformers independently and in parallel, reaching high training efficiency and stability. This also enables the application of PowerGrow in individual tasks, e.g., power grid synthesis given bus loads, or bus load generation given power grid. With this decomposition, we only need to finetune f_ϕ with LSTM-AE alternatively in the third stage of PowerGrow.

Based on the three-level graph diffusion model, the reverse diffusion process is formulated as

$$\begin{aligned}
p_\theta(A_{k-1}|A_k) &= \frac{1}{1-A_k}\texttt{Beta}\left(\frac{A_{k-1}-A_k}{1-A_k}\middle|\eta_A(\alpha_{k-1}-\alpha_k)\hat{A}_0, \eta_A(1-\alpha_{k-1}\hat{A}_0)\right) \\
p_\theta(X_{k-1}|X_k) &= \frac{1}{1-X_k}\texttt{Beta}\left(\frac{X_{k-1}-X_k}{1-X_k}\middle|\eta_X(\alpha_{k-1}-\alpha_k)\hat{X}_0, \eta_X(1-\alpha_{k-1}\hat{X}_0)\right) \\
p_\gamma(E_{k-1}|E_k) &= \frac{1}{1-E_k}\texttt{Beta}\left(\frac{E_{k-1}-E_k}{1-E_k}\middle|\eta_A(\alpha_{k-1}-\alpha_k)\hat{E}_0, \eta_A(1-\alpha_{k-1}\hat{E}_0)\right) \qquad (11) \\
p_\phi(H_{k-1}|H_k) &= \frac{1}{1-H_k}\texttt{Beta}\left(\frac{H_{k-1}-H_k}{1-H_k}\middle|\eta_X\left(\alpha_{k-1}-\alpha_k\right)\hat{H}_0, \eta_X\left(1-\alpha_{k-1}\hat{H}_0\right)\right) \\
(\hat{A}_0, \hat{X}_0) &= f_\theta(A_k, X_k, k),\ \hat{E}_0 = f_\gamma\left(\hat{A}_0, \hat{X}_0, E_k, k\right), \\
\hat{H}_0 &= f_\phi\left(\hat{A}_0, \hat{X}_0, \hat{E}_0, H_k, k\right).
\end{aligned}$$

Note that $\hat{A}_0, \hat{X}_0, \hat{E}_0$ are treated as estimation of ground truth A_0, X_0, E_0 as those ground truth are not available in generation process.

4.3 PowerGrow Framework

We employ the KLUB loss [24, 48] for training graph beta diffusion module. Take X as an example, KLUB loss is consists of two parts, time reversal loss $\mathcal{L}_{\text{sampling}}$ between $p_\theta(X_{t-1}|X_t)$ and $q(X_{t-1}|X_t, X_0)$, and error accumulation control loss $\mathcal{L}_{\text{correction}}$ between $q(X_t|\hat{X}_0)$ and $q(X_t|X_0)$.

$$\begin{aligned}
\mathcal{L}_{\text{sampling}}(X_0, t) &= \mathbb{E}_{q(X_t, X_0)}\text{KL}\left(p_\theta(X_{t-1}|X_t)\|q(X_{t-1}|X_t, X_0)\right) \\
\mathcal{L}_{\text{correction}}(X_0, t) &= \mathbb{E}_{q(X_t, X_0)}\text{KL}\left(q(X_t|\hat{X}_0)\middle\|q(X_t|X_0)\right) \\
\mathcal{L} &= \sum_{t=2}^{T}(1-\omega)\mathcal{L}_{\text{sampling}}(X_0, t) + \omega\mathcal{L}_{\text{correction}}(X_0, t). \qquad (12)
\end{aligned}$$

The time traversal loss optimizes the trainable graph transformers to match the ancestral sampling distribution with the ground truth conditional posterior distribution; the error accumulation control loss corrects the bias on the marginal distribution accumulated through the time steps. [48] derives the KL-divergence between beta distributions as Bregman divergence between log-beta distributions. Specifically, the loss terms can be expressed as

$$\begin{aligned}
\mathcal{L}_{\text{sampling}}(X_0, t) &= \mathbb{E}_{q(X_t, X_0)}\left[D_{\ln\texttt{Beta}(a,b)}\{[\eta_X(\alpha_{t-1}-\alpha_t)X_0, \eta_X(1-\alpha_{t-1}X_0)], [\eta_X(\alpha_{t-1}-\alpha_t)\hat{X}_0, \eta_X(1-\alpha_{t-1}\hat{X}_0)]\}\right] \\
\mathcal{L}_{\text{correction}}(X_0, t) &= \mathbb{E}_{q(X_t, X_0)}\left[D_{\ln\texttt{Beta}(a,b)}\{[\eta_X\alpha_t X_0, \eta_X(1-\alpha_t X_0)], [\eta_X\alpha_t\hat{X}_0, \eta_X(1-\alpha_t\hat{X}_0)]\}\right] \qquad (13)
\end{aligned}$$

where $D_f\{:,:\}$ is the Bregman divergence.

The overall training and sampling pipeline are summarized in Alg. 1 and Alg. 2 in appendix. The pipeline is also visualized in Fig.1.

In detail, we first train the LSTM Autoencoder module, then we train the three graph transformers (graph beta diffusion module), which could be done in parallel. The LSTM-AE is frozen for the cold-start stage of training f_ϕ, but in later stages, we also alternatively finetune the decoder in the LSTM Autoencoder. During Sampling process, A_0 and X_0, E_0, H_0, are sampled sequentially. Then the time series embedding H_0 is decoded with LSTM-AE. Lastly, A, X, E, D are restored to the input space.

Compared with previous diffusion works that diffuse only in raw data space [24] or latent space [32], we achieve a mixture of diffusion spaces via the three-level diffusion model. Also, compared with [32] that fixes the autoencoders for diffusion, we further align the embedding space of the autoencoder with the latent space of diffusion by finetuning the decoder along with the training of the diffusion model.

5 Experiments

5.1 Settings

Dataset preparation. To train and evaluate our proposed diffusion model for generating feasible power grid topologies, we consider two benchmark systems: the IEEE 14-bus transmission grid (U.S.) and the European 36-bus distribution grid representing an urban area [25]. For each grid, we generate 1,000 perturbed grid instances using a random-walk-based procedure. Starting from the base grid, we apply degree-preserving edge swaps to change the topology while maintaining node degree distribution. Then, we perform short random walks to extract subgraphs, which are merged back into the perturbed graph to introduce additional local structural variation. Finally, we slightly perturb line impedance values. This process yields diverse grid variants for training. Of the 1,000 generated instances, 878 variants of the 14-bus system and 813 variants of the 36-bus system are *feasible*, which means they converge under power flow simulation. The remaining instances are *infeasible* due to issues such as grid islanding or excessively high branch impedances. Only the feasible variants are used for training the proposed model. Power flow simulations are performed using the PYPOWER library [31], incorporating realistic hourly load profiles from Duquesne Light Company (Pittsburgh). This ensures that the feasibility criterion reflects real-world operational demands, as a feasible topology must be capable of serving realistic load demand patterns. To capture the increasingly dynamic and unpredictable nature of power systems, we also simulate a wide range of structural and load variability across the dataset as case studies to evaluate whether the generated grids can sustain reliable, cost-effective, and resilient performance under diverse operational scenarios.

Baselines. We compared with four state-of-the-art graph diffusion models that can deal with numerical values, including

- EDP-GNN [27] designs a permutation invariant, multi-channel graph neural network to model the score function for score matching.
- GDSS [18] is a novel continuous-time score-based generative model that model the joint distribution of nodes and edges through a system of stochastic differential equations.
- GruM [17] utilizes a mixture of diffusion processes in denoising diffusion for a rapid convergence.
- GBD [24] applies beta diffusion to graph data to model both continuous and discrete components in graphs.
- Grids generated by random-walk perturbation that we used for training and evaluation.

Metrics. This work is motivated by the need for synthetic grid datasets that exhibit both *statistical fidelity* and *operational feasible* given that real-industry grid data is often scarce, privacy-sensitive, and subject to regulatory constraints. (1) *Fidelity* assesses whether generated topologies and load profiles statistically resemble those in real-world systems. To this end, we report Maximum Mean Discrepancy (MMD) scores across multiple dimensions: the **Deg.**, **Clus.**, **Orbit**, and **Spec.** metrics correspond to MMDs over degree distributions, clustering coefficients, graphlet orbit counts, and spectral densities, respectively. **Time.** and **Attr.** evaluates MMD over time-series bus-level load and edge attributes respectively. Lower MMD values across all metrics indicate closer alignment with the empirical distribution of the training set, and thus stronger realism. (2) *Feasibility* captures whether the topology admits a convergent power flow solution with minimal violations. We quantify this using the **convergence rate** and a **feasibility score** defined as

$$\text{score} = \exp\left(-\tau \cdot \|\mathbf{v}\|_1\right) \in (0, 1], \tag{14}$$

where $\mathbf{v}$ captures violations of physical and operational constraints, such as voltage magnitude bounds, thermal branch limits, and other system inequalities [31]. A complete specification of the constraint terms is provided in Appendix A. $\tau = 10^{-5}$ is a scaling factor controlling the sensitivity to violation magnitude; the exponential decay ensures scores near 1 for nearly feasible grids and near 0 for severely infeasible ones.

Additional implementation details and code are provided in Appendix B.

5.2 Results and Analysis

We begin by visualizing representative grid instances in Figure 2. The left panel displays samples from the training set of the 14-bus and 36-bus systems, while the right shows grids generated by our model. Both sets exhibit fully connected topologies without isolated components, satisfying a fundamental requirement for operational power grids. In addition, the generated graphs, similar to those in the training set, exhibit small cycles and meshed substructures that are characteristic of urban distribution networks [42], as opposed to trivial tree-like configurations.

To assess operational feasibility, Figure 3 compares convergence rates and feasibility scores across two sets: (i) 1,000 random-walk perturbations of the 36-bus system (813 used for training), and (ii) 200 samples from our model. Perturbed variants achieve only an 81.3% convergence rate and a 0.696 average feasibility score, highlighting the difficulty of preserving physical operability. In contrast, PowerGrow attains a 98.9% convergence rate and a 0.967 average score, outperforming both perturbed variants and many training instances. Notably, feasibility is not enforced during training: our diffusion model implicitly learns to generate valid topologies by capturing structural and load patterns from data. These results demonstrate PowerGrow's ability to synthesize novel yet operationally meaningful grids, addressing a core limitation of prior

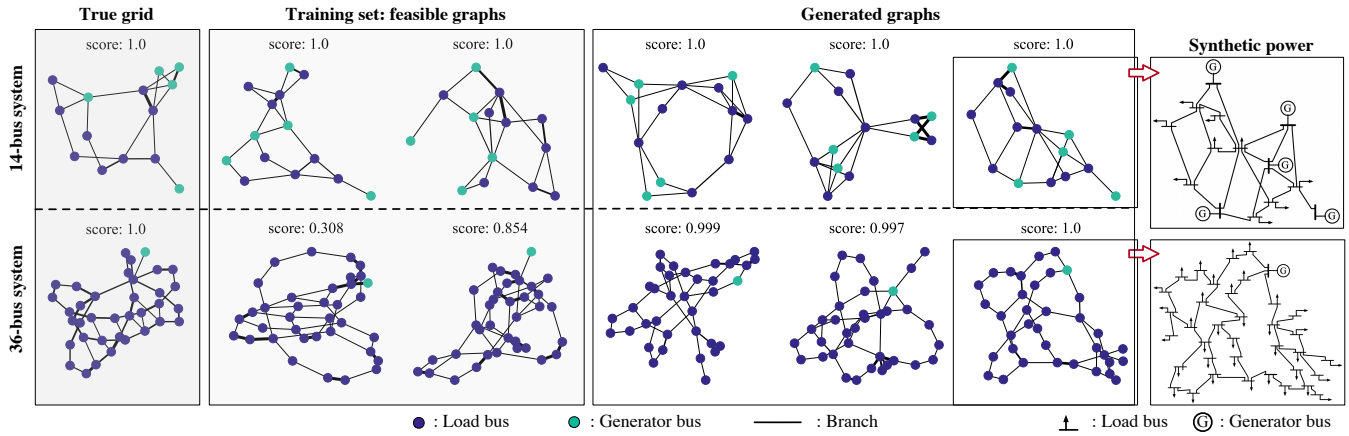


Figure 2: Visualization of generated 14-bus and 36-bus power grid samples. Nodes are color-coded by type (generator bus or load bus), and branch impedance is represented via line thickness.

Table 1: Comparison of comprehensive metrics for graph generations across baseline methods in 36-bus system.

Methods	Deg. ↓	Clus. ↓	Orbit ↓	Spec. ↓	Time. ↓	Attr. (%) ↓	Convergence rate (%) ↑	Feasibility score ↑
EDP-GNN [27]	0.4974	0.0791	0.1944	0.3677	0.0883	0.0833	0	0
GruM [17]	0.1842	0.1463	0.1089	0.1783	0.0897	0.0198	0	0
GBD [24]	0.5464	0.3053	0.2264	0.3429	0.2167	0.0136	0	0
GDSS [18]	0.0000	0.0078	0.0000	0.0085	0.0896	0.0018	67.5	0.664
Random-walk	0.0000	0.0036	0.0001	0.0033	0.0042	3×10^{-5}	81.3	0.696
PowerGrow (Ours)	0.0006	0.0037	0.0018	0.0061	0.0891	0.0077	98.9	0.967

methods and enabling realistic data generation for downstream tasks such as forecasting, control, and anomaly detection.

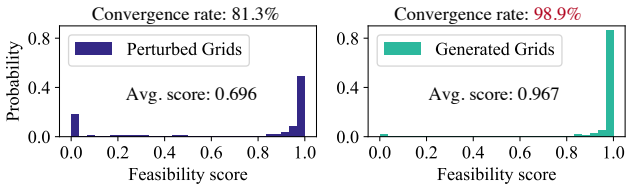


Figure 3: Comparison of convergence rate and feasibility scores in the 36-bus system dataset.

Next, we present a comprehensive evaluation in Table 1, comparing our method against a range of strong baselines. Our model achieves the highest convergence rate and feasibility score, substantially outperforming prior diffusion-based approaches such as GBD [24] and GruM [17], both of which fail to generate physically viable topologies. This failure may stem from their reliance on simultaneously learning bus types, edge structures, and attributes without explicitly modeling their interdependencies. In contrast, our hierarchical design enables effective decomposition of this complex generation process, resulting in a better balance between physical validity and statistical realism. Notably, our method achieves consistently low MMD scores across all topological and feature-level metrics, indicating strong alignment with the training distribution. While the random-walk baseline achieves near-zero MMDs by construction, it lacks physical grounding, as reflected in its degraded feasibility metrics. These results underscore the value of incorporating structured generation stages into graph diffusion models for power system synthesis.

We then evaluate the quality of the generated time-series load profiles. The left panel of Figure 4 visualizes the active and reactive power time-series generated by our model for each load bus. The active power represents real electricity consumption, while the reactive power is essential for voltage regulation and system stability. As shown in the figure, the generated profiles exhibit realistic temporal variability and oscillatory behavior, suggesting that they are sufficiently rich to support downstream tasks such as forecasting, control, and stability analysis.

To further assess the realism of these profiles, we apply Principal Component Analysis (PCA) for dimensionality reduction, followed by t-SNE to embed the high-dimensional power flow trajectories into two dimensions for visualization. We compare the resulting embeddings across three sets: time-series profiles from the training set, from our generated grids, and from infeasible grids produced via random-walk perturbations. The right panel of Figure 4 shows that the generated profiles cluster tightly with those of the feasible training set, indicating strong alignment in both structural and dynamic characteristics. In contrast, the infeasible grids form a dispersed and distinct cluster, highlighting the instability and inconsistency of their simulated power behavior. These results demonstrate that our model not only generates topologically valid grids but also

supports realistic, high-fidelity load dynamics, which are essential for practical deployment.

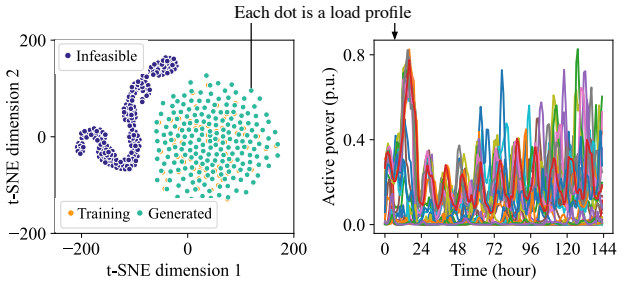


Figure 4: Left: t-SNE visualization of generated load profile latents in the 36-bus system. Right: bus-level time-series active power injections.

We also compare the sampling efficiency of PowerGrow against baselines, each tasked with generating a complete power grid sample including both the topology and corresponding load profile. Table 2 reports the average sampling time per grid, measured over 200 runs on the 36-bus system. Among the baselines, PowerGrow achieves efficient end-to-end generation of both physically feasible topologies and dynamic load profiles in 0.525s per sample. This represents up to a 31% speed improvement over baseline diffusion models such as EDP-GNN and GruM. Compared to the optimization-based method [46], which solves a constrained power flow problem to derive feasible loads, PowerGrow is over 85× faster.

Table 2: Average sampling time per grid (topology + load).

Method	Time per sample (sec)
PowerGrow (Ours)	0.525
EDP-GNN [27]	0.635
GruM [17]	0.765
GDSS [18]	2.620
GBD [24]	0.501
OPT-based [46]	46.255

5.3 Ablation Study

To assess the effectiveness of the proposed hierarchical generation strategy in PowerGrow, Figure 5 compares the generated topologies from our full model and a variant that removes the hierarchical generation structure. In the variant, all components (node types, topology, and edge attributes) are generated jointly in a single diffusion process without enforcing causal ordering. Despite being trained for the same number of epochs (60,000), this baseline produces a fragmented network with only 3 branches across 36 buses, lacking basic connectivity. In contrast, our hierarchical approach yields a well-structured and fully connected grid. This result reinforces our design choice: by decomposing the generation process into causally structured stages, our model aligns with the physical generation logic of power systems and avoids complex distribution learn. In contrast, the joint-generation variant lacks this guidance, making it prone to disconnected topologies due to the compounded difficulty of learning multiple interdependent factors simultaneously.

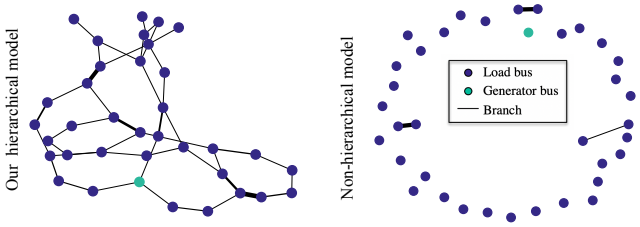


Figure 5: Generated topologies of our hierarchical diffusion model and an ablated version that applies non-hierarchical diffusion model in the 36-bus system.

5.4 Trajectory of Generated Grid Topology

To better understand the convergence behavior of our proposed diffusion model, we visualize the generated topologies at different stages of the training process. Specifically, we plot the graph topology generated by the model every 1200 training epochs Figure 6. Load profile generation trajectory is shown in Appendix C.

The early-stage generation (e.g., epoch = 1200) often results in disconnected graphs with multiple isolated buses or "islands". Such disconnected structures are undesirable for power grid planning, as they indicate infeasible networks. As training progresses, the model learns to generate more coherent and physically plausible topologies. By later epochs (e.g., epoch = 9600 and beyond), the generated graphs exhibit fully connected and structurally coherent layouts, consistently achieving feasibility scores of 1. This indicates convergence toward operationally valid grid topologies. Notably, this evolution suggests that the diffusion model progressively internalizes structural and physical constraints, despite the absence of any explicit connectivity enforcement during generation.

5.5 Case Study: Economic Efficiency

We assess the real-world value of the generated power grid topologies using a case study based on AC Optimal Power Flow (ACOPF) analysis. Each graph is assessed under a fixed load demand and generator setup, with the resulting objective cost (reflecting total generation expense) serving as a proxy for economic efficiency. We compare ACOPF cost distributions of topologies produced by our diffusion model to those from a random walk baseline, which perturbs a reference grid via stochastic edge modifications. As illustrated in Figure 7, diffusion-generated topologies consistently yield lower costs. Notably, many of the topologies discovered by the diffusion model also outperform the reference true grid. This demonstrates the model's capacity to traverse topological regions and discover more efficient power dispatches. Such capability is especially valuable for grid planning and expansion, where minimizing long-term operational cost is paramount.

5.6 Case Study: N-1 Contingency Resilience

Another critical characteristic of a plausible power grid is its ability to maintain operational stability under component failures. To

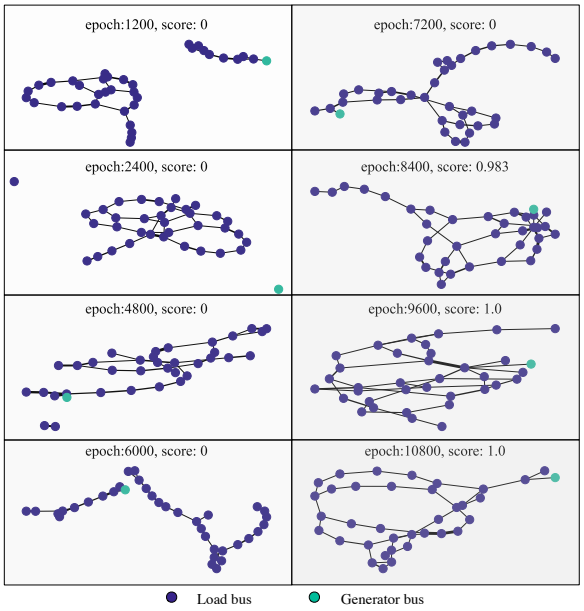


Figure 6: Evolution of generated topology across training epochs in the 36-bus dataset.

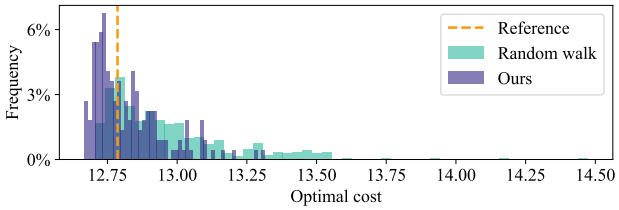


Figure 7: Histogram of ACOPF costs for grids generated by our diffusion model and random-walk procedure. The vertical dashed line is the cost of the true 36-bus system.

evaluate this property, we assess the resilience of each generated topology under the N-1 contingency criterion, which requires the system to remain feasible following the failure of any single transmission line. For each graph, we simulate the power outage of every line by disabling it and solving the resulting AC power flow. We evaluate the *N-1 resilience rate*: the proportion of single-line contingencies under which power flow converges successfully. As shown in Table 3, our diffusion-based model achieves a resilience rate of 73.19%, surpassing both the true 36-bus system (72.91%) and random walk baselines (65.24%). This demonstrates PowerGrow's ability to generate topologies that are structurally robust, as an essential property for reliable grid planning.

Table 3: N-1 resilience rate (%) across different methods on the 36-bus dataset.

Methods	Reference Grid	Random Walk	Ours
Rate (%)	72.91	65.24	73.19

5.7 Case Study: Load Shedding Under Stress

We further evaluate the stress resilience of the generated topologies, motivated by real-world scenarios such as high-demand conditions or partial generation loss. Specifically, we scale the original load demand of 14-bus system by a factor $\rho > 1$ to measure the minimum fraction of total demand that must be shed to achieve a feasible AC power flow. For each topology, we iteratively reduce load at high-demand buses until convergence is restored, using the resulting shed fraction as a measure of the system's ability to maintain service under stress. Figure 8 shows the fraction of load shed as a function of ρ for the reference grid, the random walk-generated grids (averaged), and the diffusion-generated grids (averaged). The random walk method begins requiring load shedding before $\rho = 2$, indicating limited structural resilience. In contrast, our diffusion-generated grids exhibit significantly improved behavior, closely tracking the reference grid up to $\rho \approx 3.5$, after which both methods begin to shed load gradually. It indicates that the proposed model generates topologies that are not only feasible and economically efficient but also structurally robust under stress.

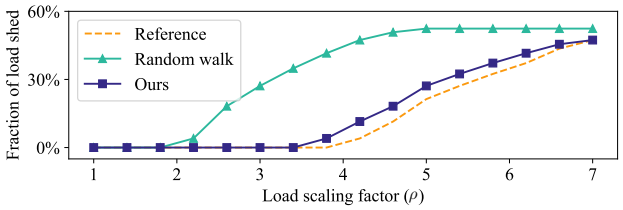


Figure 8: Fraction of load shed against load scaling factor ρ in the 14-bus dataset.

6 Conclusion and Future Work

In this work, we introduce PowerGrow, a novel co-generation framework for power grid generation and power loads synthesis. To the best of our knowledge, we are the first to leverage their inherent interdependency for the joint generation of power grid structure and load profiles with neural networks. For efficient time-series generation, we propose a three-stage training paradigm that trains a LSTM autoencoder to project time series data into a latent space and aligns the latent space with diffusion output space via finetuning. To bypass the joint generation task of heterogeneous features in power grids, we propose a three-level hierarchical graph diffusion framework for diffusion in the mixture of data and latent space. Our experiments demonstrate the high fidelity and feasibility of our generated power grids. In the future, we will automate feasibility evaluation and encourage efficient explorations using reinforcement learning or large language models.

Acknowledgments

This work is supported by NSF (2324770, 2048288) and AFOSR (FA9550-22-1-0294). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Adam B Birchfield, Kathleen M Gegner, Ti Xu, Komal S Shetye, and Thomas J Overbye. 2016. Statistical considerations in the creation of realistic synthetic power grids for geomagnetic disturbance studies. *IEEE Transactions on Power Systems* 32, 2 (2016), 1502–1510.

[2] Adam B Birchfield, Ti Xu, Kathleen M Gegner, Komal S Shetye, and Thomas J Overbye. 2016. Grid structural characteristics as validation criteria for synthetic networks. *IEEE Transactions on power systems* 32, 4 (2016), 3258–3265.

[3] Fankun Bu, Yuxuan Yuan, Zhaoyu Wang, Kaveh Dehghanpour, and Anne Kimber. 2019. A time-series distribution test system based on real utility data. In *2019 North American Power Symposium (NAPS)*. IEEE, 1–6.

[4] Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. 2023. Efficient and Degree-Guided Graph Generation via Discrete Diffusion Modeling. In *International Conference on Machine Learning*. PMLR, 4585–4610.

[5] Chandra Sekhar Charan Dande, Luca Mattorolo, Joel da Silva Andre, Lydia Lavecchia, Nikolaos Efkarpidis, and Damiano Toffanin. 2024. Synthetic Grid Generator: Synthesizing Large-Scale Power Distribution Grids using Open Street Map. *arXiv preprint arXiv:2408.04923* (2024).

[6] Alexander Eigeles Emanuel. 2011. *Power definitions and the physical mechanism of power flow*. John Wiley & Sons.

[7] Joaquín Delgado Fernández, Sergio Potenciano Menci, Chul Min Lee, Alexander Rieger, and Gilbert Fridgen. 2022. Privacy-preserving federated learning for residential short-term load forecasting. *Applied energy* 326 (2022), 119915.

[8] David Fischer, Arne Surmann, and Karen Byskov Lindberg. 2020. Impact of emerging technologies on the electricity load profile of residential areas. *Energy and Buildings* 208 (2020), 109614.

[9] Francesco Giacomarra, Gianmarco Bet, and Alessandro Zocca. 2024. Generating synthetic power grids using exponential random graph models. *PRX Energy* 3, 2 (2024), 023005.

[10] Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45.

[11] Zhiye Guo, Jian Liu, Yanli Wang, Mengrui Chen, Duolin Wang, Dong Xu, and Jianlin Cheng. 2024. Diffusion models in bioinformatics and computational biology. *Nature reviews bioengineering* 2, 2 (2024), 136–154.

[12] Tyler Hodge. 2020. Hourly electricity consumption varies throughout the day and across seasons. *US Energy Information Administration* 21 (2020).

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

[14] Yi Hu, Yiyan Li, Lidong Song, Han Pyo Lee, PJ Rehm, Matthew Makdad, Edmond Miller, and Ning Lu. 2023. MultiLoad-GAN: A GAN-based synthetic load group generation method considering spatial-temporal correlations. *IEEE Transactions on Smart Grid* 15, 2 (2023), 2309–2320.

[15] Chenqing Hua, Sitao Luan, Minkai Xu, Zhitao Ying, Jie Fu, Stefano Ermon, and Doina Precup. 2024. Mudiff: Unified diffusion for complete molecule generation. In *Learning on Graphs Conference*. PMLR, 33–1.

[16] Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Conditional diffusion based on discrete graph structures for molecular graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 4302–4311.

[17] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. 2024. Graph Generation with Diffusion Mixture. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 22371–22405. https://proceedings.mlr.press/v235/jo24b.html

[18] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*. PMLR, 10362–10383.

[19] Filip Ekström Kelvinius, Oskar B Andersson, Abhijith S Parackal, Dong Qian, Rickard Armiento, and Fredrik Lindsten. [n. d.]. WyckoffDiff–A Generative Diffusion Model for Crystal Symmetry. In *Forty-second International Conference on Machine Learning*.

[20] Mahdi Khodayar, Jianhui Wang, and Zhaoyu Wang. 2019. Deep generative graph distribution learning for synthetic power grids. *arXiv preprint arXiv:1901.09674* (2019).

[21] Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. 2023. Autoregressive diffusion model for graph generation. In *International conference on machine learning*. PMLR, 17391–17408.

[22] Shimiao Li, Jan Drgona, Shrirang Abhyankar, and Larry Pileggi. 2023. Power grid behavioral patterns and risks of generalization in applied machine learning. In *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*. 106–114.

[23] Gang Liu, Jiaxin Xu, Tengfei Luo, and Meng Jiang. 2024. Graph diffusion transformers for multi-conditional molecular generation. *Advances in Neural Information Processing Systems* 37 (2024), 8065–8092.

[24] Xinyang Liu, Yilin He, Bo Chen, and Mingyuan Zhou. 2025. Advancing Graph Generation through Beta Diffusion. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=x1An5a3U9I

[25] Carlos Mateo, Giuseppe Prettico, Tomás Gómez, Rafael Cossent, Flavia Gangale, Pablo Frías, and Gianluca Fulli. 2018. European representative electricity distribution networks. *International Journal of Electrical Power & Energy Systems* 99 (2018), 273–280.

[26] Daniel K Molzahn, Florian Dörfler, Henrik Sandberg, Steven H Low, Sambuddha Chakrabarti, Ross Baldick, and Javad Lavaei. 2017. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Transactions on Smart Grid* 8, 6 (2017), 2941–2962.

[27] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. 2020. Permutation Invariant Graph Generation via Score-Based Generative Modeling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 108)*, Silvia Chiappa and Roberto Calandra (Eds.). PMLR, 4474–4484. https://proceedings.mlr.press/v108/niu20a.html

[28] Andrea Pinceti, Oliver Kosut, and Lalitha Sankar. 2019. Data-driven generation of synthetic load datasets preserving spatio-temporal features. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 1–5.

[29] Andrea Pinceti, Lalitha Sankar, and Oliver Kosut. 2021. Synthetic time-series load data via conditional generative adversarial networks. In *2021 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 1–5.

[30] Elisavet Proedrou. 2021. A comprehensive review of residential electricity load profile models. *IEEE Access* 9 (2021), 12114–12133.

[31] R. Richards. 2013. PYPOWER: Electric power system simulation in Python. https://github.com/rwl/PYPOWER. Version 5.0.

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[33] Mohammad Shahraeini. 2023. Modified Erdős–Rényi Random Graph Model for Generating Synthetic Power Grids. *IEEE Systems Journal* 18, 1 (2023), 96–107.

[34] Yulin Song, Han Yan, Chenxi Sun, and Jianwei Huang. 2025. Synthetic Power Network Topology Generation with Geographical Information. In *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*. 712–718.

[35] Yujie Tang, Krishnamurthy Dvijotham, and Steven Low. 2017. Real-time optimal power flow. *IEEE Transactions on Smart Grid* 8, 6 (2017), 2963–2973.

[36] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2023. DiGress: Discrete Denoising diffusion for graph generation. In *ICLR*.

[37] Dominik JE Waibel, Ernst Röell, Bastian Rieck, Raja Giryes, and Carsten Marr. 2023. A diffusion model predicts 3d shapes from 2d microscopy images. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 1–5.

[38] Chenguang Wang, Simon H Tindemans, and Peter Palensky. 2022. Generating contextual load profiles using a conditional variational autoencoder. In *2022 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, 1–6.

[39] Yi Wang, Qixin Chen, Tao Hong, and Chongqing Kang. 2018. Review of smart meter data analytics: Applications, methodologies, and challenges. *IEEE Transactions on smart Grid* 10, 3 (2018), 3125–3148.

[40] Allen J Wood, Bruce F Wollenberg, and Gerald B Sheblé. 2013. *Power generation, operation, and control*. John wiley & sons.

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[42] Chenhan Xiao, Yizheng Liao, and Yang Weng. 2023. Distribution Grid Line Outage Identification with Unknown Pattern and Performance Guarantee. *IEEE Transactions on Power Systems* (2023).

[43] Tian Xie, Xiang Fu, Octavian-Eugen Ganea, Regina Barzilay, and Tommi S Jaakkola. [n. d.]. Crystal Diffusion Variational Autoencoder for Periodic Material Generation. In *International Conference on Learning Representations*.

[44] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. 2024. Discrete-state Continuous-time Diffusion for Graph Generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=YkSKZEhIYt

[45] Rong Yan, Yuxuan Yuan, Zhaoyu Wang, Guangchao Geng, and Quanyuan Jiang. 2021. Synthetic Active Distribution System Generation via Unbalanced Graph Generative Adversarial Network. *arXiv preprint arXiv:2108.00599* (2021).

[46] Rong Yan, Yuxuan Yuan, Zhaoyu Wang, Guangchao Geng, and Quanyuan Jiang. 2022. Active distribution system synthesis via unbalanced graph generative adversarial network. *IEEE Transactions on Power Systems* 38, 5 (2022), 4293–4307.

[47] Sherry Yang, KwangHwan Cho, Amil Merchant, Pieter Abbeel, Dale Schuurmans, Igor Mordatch, and Ekin Dogus Cubuk. [n. d.]. Scalable Diffusion for Materials Generation. In *The Twelfth International Conference on Learning Representations*.

[48] Mingyuan Zhou, Tianqi Chen, Zhendong Wang, and Huangjie Zheng. 2023. Beta diffusion. *Advances in Neural Information Processing Systems* 36 (2023), 30070–30095.

[49] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. 2010. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems* 26, 1 (2010), 12–19.

A Construction of the Violation Vector

To quantify the physical plausibility of generated power grids, we define a continuous feasibility score based on the magnitude of operational constraint violations. Let the result of an AC power flow simulation be represented by voltage magnitudes V_i, phase angles θ_i, and complex power flows $S_{ij} = P_{ij} + jQ_{ij}$ on branch (i, j). The constraint violation vector $\mathbf{v}$ aggregates violation magnitudes for key physical constraints:

- Voltage magnitude limits. For each bus i, voltage must remain within its permissible bounds: $V_i^{\min} \le V_i \le V_i^{\max}$, which indicates the constraint violation as
$$v_i^{(V)} = \max\left(0, V_i - V_i^{\max}\right) + \max\left(0, V_i^{\min} - V_i\right). \quad (15)$$
- Branch flow limits. Each transmission line must respect its thermal constraint on apparent power: $|S_{ij}| \le S_{ij}^{\max}$, which indicates the constraint violation as
$$v_{ij}^{(S)} = \max\left(0, |S_{ij}| - S_{ij}^{\max}\right). \quad (16)$$
- Generator capacity limits. When available, generator outputs must lie within capacity constraints: $P_i^{\min} \le P_i^{\text{gen}} \le P_i^{\max}$ and $Q_i^{\min} \le Q_i^{\text{gen}} \le Q_i^{\max}$, which indicates the constraint violation as
$$v_i^{(P)} = \max\left(0, P_i^{\text{gen}} - P_i^{\max}\right) + \max\left(0, P_i^{\min} - P_i^{\text{gen}}\right), \quad (17)$$
$$v_i^{(Q)} = \max\left(0, Q_i^{\text{gen}} - Q_i^{\max}\right) + \max\left(0, Q_i^{\min} - Q_i^{\text{gen}}\right). \quad (18)$$

The complete violation vector is given by:

$$\mathbf{v} = \left[\{v_i^{(V)}\}_i, \{v_{ij}^{(S)}\}_{(i,j)}, \{v_i^{(P)}\}_i, \{v_i^{(Q)}\}_i\right],$$

where generator terms $v^{(P)}$ and $v^{(Q)}$ are included only if generator limits are available in the dataset. In experiments, the vector $\mathbf{v}$ is constructed using the raw constraint violation array from the `runopf` routine in PYPOWER [31], accessed via `results["raw"]["g"]`. This formulation enables a differentiable and fine-grained feasibility score, facilitating smooth evaluation of physical realism in generated grids.

B Implementation Details

Our codes and dataset are available at https://github.com/xinyuuhe/PowerGrow. The overall training and sampling pipelines are summarized in Alg. 1 and Alg. 2.

Data Preprocessing. We normalize the branch attributes to bounded range $[0, 1]$, for example, for branch resistances in the raw dataset $\{\tilde{r}|\tilde{r} \in \mathbb{R}^M\}$, where M is the number of branches in each graph, it is normalized by

$$r = (\tilde{r} - \mu_r) * \sigma_r + 0.5 \quad (19)$$

where μ_r, σ_r are predefined parameters so that $\mu_r = mean_{\tilde{r}}\big(mean(\tilde{r})\big)$, and $max(r)$ is close or equal to 1. Time-series load profile are also rescaled, e.g., for real-part of load profiles $\{\tilde{P}|\tilde{P} \in \mathbb{R}^{N\times T}\}$, we apply $P = \tilde{P} \cdot \sigma_P$ so that $max(P)$ is close to 1.

Logit domain computation. To ensure numerical accuracy, we follow [24, 48] to train in the logit space. For example, in logit domain, Eq. (3) is expressed as

$$\text{logit}(A_{k-1}) = \ln\left(e^{\text{logit}(A_k)} + e^{\text{logit}(P_{A,k})} + e^{\text{logit}(A_k)+\text{logit}(P_{A,k})}\right). \quad (20)$$

Baselines. We run their official implementations for baselines. GruM, EDP-GNN, GDSS only generate one-dimensional edge feature (adjacency matrix) in their original version, therefore we extend the graph transformers in their codes to accept multi-dimensional features by considering the dimensions as different channels and copying the operations on each dimension. EDP-GNN does not generate node attributes as well, therefore we randomly generate load profiles through our pretrained decoder in LSTM-AE with random noise as inputs. As EDP-GNN reaches the lowest Time. MMD in all baselines, the effectiveness of our LSTM-AE for time-series generation is further confirmed.

C Evolution of Load Profile Generation During Diffusion Training

In addition to topology evolution, we also investigate how the generated time-series load profiles improve over the training of diffusion training. We focus on the real power consumption vector P_{vec}, expressed in per unit (p.u.), and visualize the generated time-series curves every 1200 epochs.

Figure 9 presents the generated P_{vec} sequences across training stages. Early in training, the generated profiles exhibit limited variability and lack meaningful temporal patterns. However, it already exhibits clear periodical patterns which confirms effectiveness of pretrained LAST-AE. As training advances, the model begins to produce more structured and realistic time-series patterns, capturing typical fluctuations found in real-world load behavior. By the final stages of training, the time-series exhibit smooth yet diverse trajectories that align with expected residential or commercial load dynamics. This progression highlights the ability of the diffusion model to learn both spatial and temporal structures jointly.

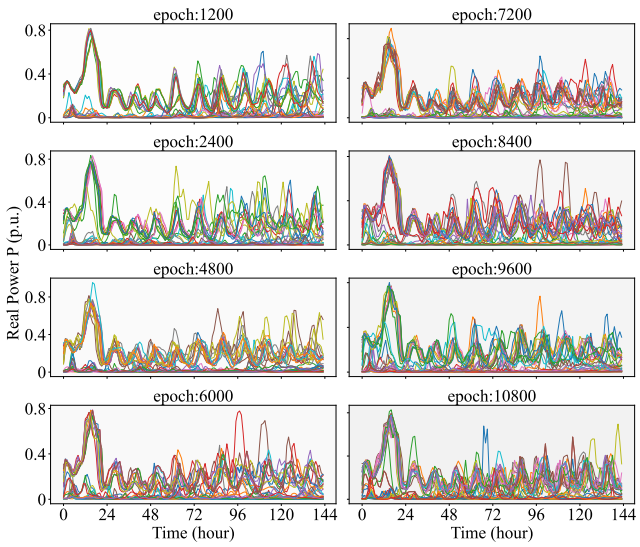


Figure 9: Evolution of generated real power (p.u.) time-series across training epochs in the 36-bus dataset.

Algorithm 2 Sampling Process of PowerGrow

Input:
Number of diffusion steps T, concentration parameters η_X and η_A, graph transformer models $f_\theta(), f_\gamma(), f_\phi()$, LSTM Autoencoder $g()$, normalization parameters $\mathbf{w} = (w_A\ w_X\ w_E\ w_D), \mathbf{b} = (w_A\ w_X\ w_E\ w_D)$

Sample $A_T \sim \texttt{Beta}(\eta_A \alpha_T \mathbb{E}[A_0]),\ E_T \sim \texttt{Beta}(\eta_A \alpha_T \mathbb{E}[E_0])$
Sample $X_T \sim \texttt{Beta}(\eta_X \alpha_T \mathbb{E}[X_0]),\ H_T \sim \texttt{Beta}(\eta_X \alpha_T \mathbb{E}[H_0])$
for t from T to 1 **do**
  $\hat{A}'_0, \hat{X}'_0 \leftarrow f_\theta(A_t, X_t, t)$
  $\hat{A}_0 \leftarrow w_A \hat{A}'_0 + b_A, \hat{X}_0 \leftarrow w_X \hat{X}'_0 + b_X$
  Sample A_{t-1}, X_{t-1} via Eq. (3)
end for
for t from T to 1 **do**
  $\hat{E}'_0 \leftarrow f_\gamma(A_0, X_0, E_t, t)$
  $\hat{E}_0 \leftarrow w_E \hat{E}'_0 + b_E$
  Sample E_{t-1} via Eq. (3)
end for
for t from T to 1 **do**
  $\hat{H}'_0 \leftarrow f_\phi(A_0, X_0, E_0, H_t, t)$
  Sample H_{t-1} via Eq. (3)
end for
$D_0 \leftarrow g_{dec}(H_0)$
Output $(A\ X\ E\ D) = \big((A_0\ X_0\ E_0\ D_0) - \mathbf{b}\big)/\mathbf{w}$

Algorithm 1 Training pipeline of PowerGrow

Input:
Number of diffusion steps T, concentration parameters η_X and η_A, graph transformer models $f_\theta(), f_\gamma(), f_\phi()$, LSTM Autoencoder $g()$, input graphs in batches $\mathbb{B}$, number of epochs E, normalization parameters $\mathbf{w} = (w_A\ w_X\ w_E\ w_D), \mathbf{b} = (w_A\ w_X\ w_E\ w_D)$, number of cold-start epochs M

for epoch from 1 to E **do**
  for $(A, X, E, D) \in \mathbb{B}$ **do**
    Normalize $D_0 \leftarrow w_D D + b_D$
    Reconstruct $\hat{D}_0 \leftarrow g(D_0)$
    Update $g()$ with $\mathcal{L} = \text{MSE}(D_0, \hat{D}_0)$
  end for
end for
for epoch from 1 to E **do**
  for $(A, X, E, D) \in \mathbb{B}$ **do**
    Normalize $(A_0\ X_0\ E_0\ D_0) \leftarrow \mathbf{w}(A\ X\ E\ D) + \mathbf{b}$
    $t \sim \text{Unif}(1, ..., T)$
    Sample A_t, X_t with Eq. (5)
    Predict $\hat{A}_0, \hat{X}_0 \leftarrow f_\theta(A_t, X_t, t)$
    Update $f_\theta()$ with KLUB loss Eq. (12) on A and X
  end for
end for
for epoch from 1 to E **do**
  for $(A, X, E, D) \in \mathbb{B}$ **do**
    Normalize $(A_0\ X_0\ E_0\ D_0) \leftarrow \mathbf{w}(A\ X\ E\ D) + \mathbf{b}$
    $t \sim \text{Unif}(1, ..., T)$
    Sample E_t with Eq. (5)
    Predict $\hat{E}_0 \leftarrow f_\gamma(A_0, X_0, E_t, t)$
    Update $f_\gamma()$ with KLUB loss Eq. (12) on E
  end for
end for
for epoch from 1 to E **do**
  for $(A, X, E, D) \in \mathbb{B}$ **do**
    Normalize $(A_0\ X_0\ E_0\ D_0) \leftarrow \mathbf{w}(A\ X\ E\ D) + \mathbf{b}$
    $t \sim \text{Unif}(1, ..., T)$
    $H_0 \leftarrow g_{enc}(D_0)$
    Sample H_t with Eq. (5)
    Predict $\hat{H}_0 \leftarrow f_\phi(A_0, X_0, E_0, H_t, t)$
    Update $f_\phi()$ with KLUB loss Eq. (12) on H
    if epoch $> M$ **then**
      $\hat{D}_0 \leftarrow g_{dec}(\hat{H}_0)$
      Update $g()$ with $\text{MSE}(D_0, \hat{D}_0)$
    end if
  end for
end for