Diffusion Models for Graph Signal Augmentation: From Noise to Realistic Mesh Dynamics

Technical Report

January 6, 2026

Abstract

This report describes the application of diffusion models to generate realistic graph signals on fixed meshes. We motivate the approach by examining the limitations of simpler methods (spectral filtering, VAEs), then develop a graph neural network-based diffusion model for trajectory generation. A key insight is that the diffusion framework naturally supports data augmentation by controlling the noise injection level, allowing generation of signals that are "similar but not identical" to training examples.

1 Introduction and Motivation

1.1 The Problem

Given a set of graph signals defined on a fixed mesh (e.g., simulation results from finite element analysis or physics simulations), we want to generate new signals that:

1. Look realistic (could plausibly come from the same physical process)
2. Are diverse (not identical to training examples)
3. Respect the underlying structure (mesh connectivity, boundary conditions)

1.2 Use Case: Data Augmentation

The primary application is **data augmentation** for training other graph neural networks. When real simulation data is expensive to generate, we want to:

- Take a small set of real examples
- Generate many similar-but-different variations
- Use these to augment training data

2 Methods We Explored

2.1 Approach 1: Spectral Noise Filtering

2.1.1 Method

The idea was to use the Windowed Graph Fourier Transform (following Shuman et al.) to:

1. Compute the spectral envelope $|S(v, \lambda)|$ of a real signal
2. Generate white noise in the spectral domain
3. Filter the noise with the spectral envelope
4. Inverse transform to get a new signal

2.1.2 Result

This approach failed to capture:

- **Boundary conditions**: Fixed vertices (e.g., flag pole) moved freely
- **Spatial coherence**: Nearby vertices should move together
- **Temporal smoothness**: Generated motion was too jittery

The spectral envelope captures only marginal statistics, not the joint structure.

2.2 Approach 2: Spectral VAE

2.2.1 Method

A variational autoencoder operating in the spectral domain:

$$x \xrightarrow{\text{GFT}} \hat{x} \xrightarrow{\text{Encoder}} \mu, \sigma \xrightarrow{z=\mu+\sigma\cdot\epsilon} \xrightarrow{\text{Decoder}} \hat{x}' \xrightarrow{\text{IGFT}} x' \tag{1}$$

Noise is injected in the latent space via the reparameterization trick.

2.2.2 Result

On steady-state FEA data (heat equation solutions), the VAE produced outputs nearly identical to inputs. The training data lacked diversity—all signals were smooth heat distributions varying only in parameters.

Key insight: Generative models need diverse training data to learn meaningful variation.

3 Diffusion Models: The Solution

3.1 Why Diffusion?

Diffusion models are the current state-of-the-art for generative AI (Stable Diffusion, DALL-E 3, Sora). They:

- Learn implicit constraints from data
- Generate high-quality, diverse outputs
- Handle high-dimensional data naturally
- Support controllable generation

3.2 The Diffusion Process

3.2.1 Forward Process (Adding Noise)

Given a clean signal x_0, we progressively add Gaussian noise:

$$x_n = \sqrt{\bar{\alpha}_n}\, x_0 + \sqrt{1 - \bar{\alpha}_n}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{2}$$

where $\bar{\alpha}_n = \prod_{i=1}^{n} \alpha_i$ is the cumulative noise schedule, and n is the **noise step** (not to be confused with simulation time t).

At $n = 0$: signal is clean (x_0).
At $n = N$: signal is pure noise ($x_N \approx \mathcal{N}(0, I)$).

3.2.2 Reverse Process (Denoising)

A neural network ϵ_θ learns to predict the noise:

$$\hat{\epsilon} = \epsilon_\theta(x_n, n) \tag{3}$$

Given $\hat{\epsilon}$, we can estimate the clean signal and take a denoising step.

3.2.3 Training

Algorithm 1 Diffusion Training

repeat
  Sample $x_0 \sim$ training data
  Sample $n \sim \text{Uniform}(1, N)$
  Sample $\epsilon \sim \mathcal{N}(0, I)$
  $x_n \leftarrow \sqrt{\bar{\alpha}_n}\, x_0 + \sqrt{1 - \bar{\alpha}_n}\, \epsilon$
  Gradient step on $\|\epsilon - \epsilon_\theta(x_n, n)\|^2$
until converged

3.2.4 Generation

Algorithm 2 Diffusion Generation

$x_N \sim \mathcal{N}(0, I)$ ▷ Start from pure noise
for $n = N, N-1, \ldots, 1$ **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_n, n)$
  $x_{n-1} \leftarrow \text{DenoisingStep}(x_n, \hat{\epsilon}, n)$
end for
return x_0 ▷ Clean generated signal

4 Key Insight: Augmentation via Partial Denoising

4.1 The "img2img" Trick

Instead of starting from pure noise, we can:

1. Take a **real** signal x_0
2. Add noise to an **intermediate** level n^*
3. Denoise from n^* down to 0

This produces a signal that is a **variation** of the original.

Algorithm 3 Signal Augmentation via Partial Denoising

Input: Real signal x_0, starting noise step n^*
Sample $\epsilon \sim \mathcal{N}(0, I)$
$x_{n^*} \leftarrow \sqrt{\bar{\alpha}_{n^*}}\, x_0 + \sqrt{1 - \bar{\alpha}_{n^*}}\, \epsilon$
for $n = n^*, n^* - 1, \ldots, 1$ **do**
    $\hat{\epsilon} \leftarrow \epsilon_\theta(x_n, n)$
    $x_{n-1} \leftarrow \text{DenoisingStep}(x_n, \hat{\epsilon}, n)$
end for
return x_0 ▷ Augmented signal

4.2 Controlling Similarity

The starting noise step n^* acts as a "similarity dial":

n^*	Noise Added	Result
N (e.g., 1000)	Maximum	Completely new signal
$0.7N$ (e.g., 700)	High	Loosely similar to original
$0.3N$ (e.g., 300)	Medium	Same structure, different details
$0.1N$ (e.g., 100)	Low	Very close to original
0	None	Identical to input

This is exactly the augmentation we originally wanted: generate signals that are similar but not identical, with controllable similarity.

5 Architecture for Mesh Data

5.1 Why Graph Neural Networks?

Mesh data is not a regular grid, so standard convolutions don't apply. We use GNNs that:

- Respect mesh connectivity (which vertices are neighbors)
- Handle irregular topology
- Encode spatial structure naturally

5.2 Model Architecture

For trajectory data ($T \times V \times 3$) where T is the number of simulation frames, V is the number of vertices, and 3 is the spatial dimension:

Component	Purpose
GNN Encoder	Process each frame, respecting mesh structure
Temporal Transformer	Model dynamics across simulation time
Noise Step Embedding	Condition on diffusion noise level n
GNN Decoder	Output denoised positions

5.3 Message Passing on Meshes

The GNN uses message passing on mesh edges:

$$h_i^{(l+1)} = h_i^{(l)} + \text{Aggregate}_{j \in \mathcal{N}(i)} \text{MLP}\left(h_i^{(l)}, h_j^{(l)}\right) \tag{4}$$

where $\mathcal{N}(i)$ is the set of vertices connected to vertex i by a mesh edge.

6 Connection to Original Goals

Original Goal	How Diffusion Achieves It
Generate similar-but-different signals	Partial denoising with controllable n^*
Augment training data	Generate many variations from few examples
Respect mesh structure	GNN encoder/decoder uses mesh connectivity
Capture temporal dynamics	Temporal transformer models sequence
Learn implicit constraints	Training on real data learns physics

7 Summary

1. **Simple approaches failed**: Spectral filtering and small VAEs couldn't capture the full structure of mesh dynamics.
2. **Diffusion learns holistically**: By learning to denoise at all noise levels, the model captures spatial coherence, temporal smoothness, and physical constraints implicitly.
3. **Augmentation is built-in**: The partial denoising trick provides controllable augmentation without any modification to the model.
4. **GNNs respect structure**: Graph neural networks naturally handle irregular mesh topology.

8 Next Steps

1. Implement the trajectory diffusion model (PyTorch + PyTorch Geometric)
2. Train on flag simulation data (DeepMind MeshGraphNets dataset)
3. Evaluate generation quality and augmentation effectiveness
4. Apply to other mesh simulation domains

References

[1] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *NeurIPS*.

[2] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. W. (2021). Learning Mesh-Based Simulation with Graph Networks. *ICLR*.

[3] Shuman, D. I., et al. (2013). The Emerging Field of Signal Processing on Graphs. *IEEE Signal Processing Magazine*.

[4] Meng, C., et al. (2022). SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. *ICLR*.