

Zero-Shot Forecasting of Network Dynamics through Weight Flow Matching

Shihe Zhou*

zhoush23@mails.tsinghua.edu.cn
Xingjian College
Tsinghua University
Beijing, China

Huandong Wang[†]

wanghuandong@tsinghua.edu.cn
Department of Electronic Engineering BNRist
Tsinghua University
Beijing, China

Ruikun Li*

lrk23@mails.tsinghua.edu.cn
Shenzhen International Graduate School
Tsinghua University
Shenzhen, China

Yong Li

liyong07@tsinghua.edu.cn
Department of Electronic Engineering BNRist
Tsinghua University
Beijing, China

Abstract

Forecasting state evolution of network systems, such as the spread of information on social networks, is significant for effective policy interventions and resource management. However, the underlying propagation dynamics constantly shift with new topics or events, which are modeled as changing coefficients of the underlying dynamics. Deep learning models struggle to adapt to these out-of-distribution shifts without extensive new data and retraining. To address this, we present Zero-Shot Forecasting of Network Dynamics through Weight Flow Matching (FNFM), a generative, coefficient-conditioned framework that generates dynamic model weights for an unseen target coefficient, enabling zero-shot forecasting. Our framework utilizes a Variational Encoder to summarize the forecaster weights trained in observed environments into compact latent tokens. A Conditional Flow Matching (CFM) module then learns a continuous transport from a simple Gaussian distribution to the empirical distribution of these weights, conditioned on the dynamical coefficients. This process is instantaneous at test time and requires no gradient-based optimization. Across varied dynamical coefficients, empirical results indicate that FNFM yields more reliable zero-shot accuracy than baseline methods, particularly under pronounced coefficient shift. Code is available: <https://github.com/tsinghua-fib-lab/FNFM>.

CCS Concepts

• **Networks** → **Network dynamics**; *Network performance modeling*; *Network simulations*; • **Applied computing** → *Law, social and behavioral sciences*; *Physical sciences and engineering*.

Keywords

Network dynamics, Multi-environment learning, Flow matching

*Both authors contributed equally to this research.

[†]Corresponding authors



This work is licensed under a Creative Commons Attribution 4.0 International License. *WWW '26, Dubai, United Arab Emirates*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
<https://doi.org/10.1145/3774904.3792722>

ACM Reference Format:

Shihe Zhou, Ruikun Li, Huandong Wang, and Yong Li. 2026. Zero-Shot Forecasting of Network Dynamics through Weight Flow Matching. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, April 13–17, 2026, Dubai, United Arab Emirates. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3774904.3792722>

1 Introduction

The propagation of behaviors, evolution of cultural norms, and even the formation of consensus on social networks can all be modeled as spatiotemporal dynamic processes among individuals on complex networks [18, 24, 50]. Recent advances in artificial intelligence have further expanded our capacity to represent and control these empirical networked systems across diverse domains, from social media to urban environments [4]. Accurately forecasting the evolution of these dynamics is crucial for understanding sociophysical phenomena and for key applications such as curbing the spread of misinformation [2, 17, 32]. However, such tasks often involve cross-scale interactions where unknown mechanisms and high computational costs necessitate the integration of domain knowledge with data-driven simulation [41]. The complexity of network dynamics stems from the intricate interplay between network topology and the parameters of the underlying dynamics. Even with identical topologies and governing equations, subtle shifts in dynamic parameters can push a system toward entirely different critical regimes, fundamentally altering its propagation behavior [9, 14, 29, 40].

As we demonstrate on a social media information propagation model (Figure 1a), a mere difference in popularity coefficients (define as $\frac{\beta}{\gamma}$) leads two propagation trajectories toward starkly different outcomes: rapid decay versus viral spread. This phenomenon precisely captures the disparity in how opinions on different topics propagate through cyberspace, while also posing a stringent challenge to the generalization capability of predictive models: they must be able to accurately forecast dynamic evolution in new environments.

To train generalizable models from a limited set of observed environments, existing work predominantly follows two paths. The first path involves building "one-for-all" spatio-temporal foundation models, attempting to train a universal predictor by aggregating

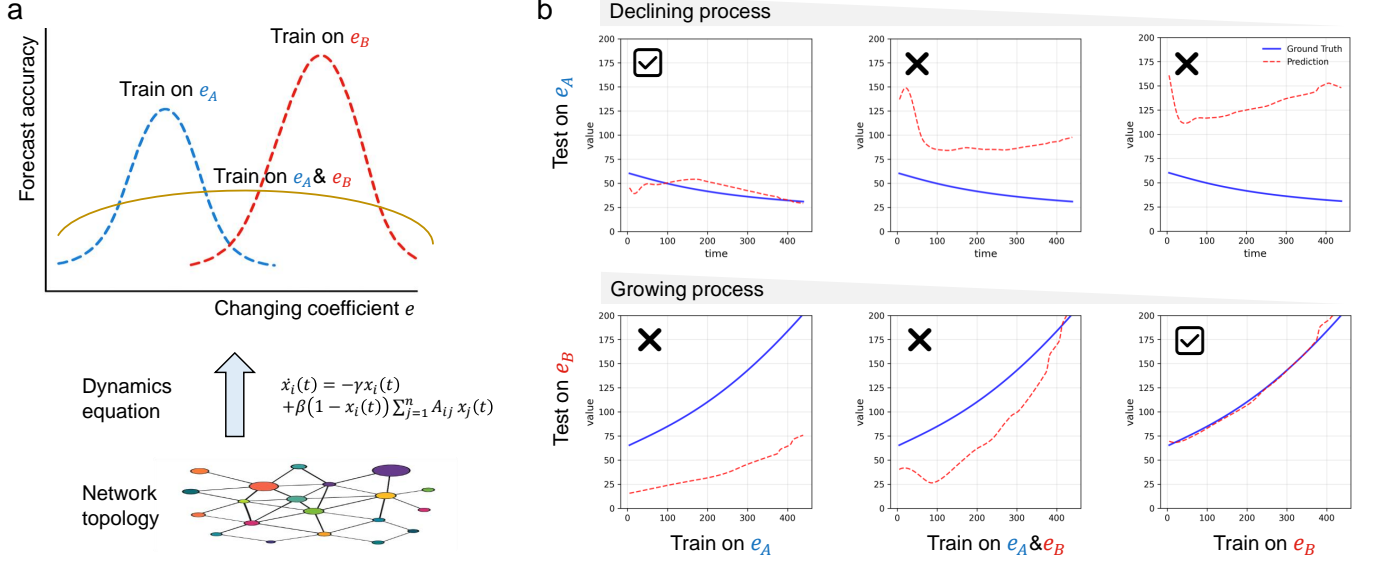


Figure 1: The generalization trap in network dynamics. (a) A generalist model trained on mixed data struggles to outperform specialized expert models. Network dynamics adopts the classic information dissemination model [40], in which the dynamic behavior is governed by the popularity coefficient. (b) Training and testing performance on cross-environment propagation dynamics, where e_A and e_B are propagation processes with different coefficients.

data from all environments [11, 25, 26, 46]. However, these monolithic models, guided by the principle of empirical risk minimization, often achieve generalization at the expense of specialized performance, leading to performance on specific tasks that can be inferior to that of much smaller expert models (as shown in Figure 1b). The second path is based on meta-learning approaches, which rapidly adapt a model to new environments, thereby reducing data dependency [33–35, 44]. Nonetheless, meta-learning frameworks still rely on the availability of at least a small amount of historical trajectory data from the target environment for finetuning. In practice, this precondition does not hold in many high-value predictive scenarios [5, 8, 13, 21]. In such scenarios, a decision-maker might need to predict the potential consequences of a hypothetical environmental coefficient (e.g., the adoption rate of a new policy). Therefore, how to perform reliable zero-shot prediction for network dynamics under new environmental coefficients remains a critical open question.

In this paper, we introduce Forecasting of Network Dynamics through Weight Flow Matching (FNFM), a novel generative framework that addresses the challenge of zero-shot prediction for network dynamics across varying environments. Instead of predicting trajectories directly, FNFM learns to generate the complete *weights* of a specialized forecaster model tailored to any given environmental coefficients. FNFM first collects a diverse set of expert weights from various seen environments. It then employs a Variational Autoencoder (VAE) to learn a compact and smooth latent manifold of these weights. Finally, a Conditional Flow Matching (CFM) model is trained to map environmental coefficients to this manifold, enabling the conditional synthesis of new latent vectors. When inferring, this process is instantaneous and requires no finetuning,

making FNFM a powerful tool for forecasting for novel scenarios on demand.

Our main contributions are summarized as follows:

- We propose a new paradigm for zero-shot forecasting of network dynamics, shifting the objective from trajectory prediction to the direct generation of model weights.
- We introduce FNFM, a novel framework that operationalizes this paradigm by synergistically combining a VAE and a Conditional Flow Matching model to learn the complex mapping from dynamic coefficients to optimal model weights.
- We conduct extensive experiments demonstrating that FNFM significantly outperforms state-of-the-art baselines by an average of 8.30% in zero-shot forecasting scenarios, showcasing its superior generalization.

2 Preliminary

2.1 Problem Definition

We consider a dynamic process evolving over a network of n nodes, where each node possesses a d -dimensional feature vector. A core challenge in forecasting such dynamics is that while different environments may share the same underlying network topology and governing equations, they are distinguished by a set of dynamic coefficients. These coefficients, denoted by an environmental vector $e \in E$, critically alter the system's behavior, leading to fundamentally different temporal evolution patterns.

Formally, given an adjacency matrix $A \in \mathbb{R}^{n \times n}$ and the environmental coefficient vector e , the network dynamics can be described by a system of ordinary differential equations (ODEs):

$$\frac{d\mathbf{X}(t)}{dt} = F(\mathbf{X}(t), A, e)$$

where $\mathbf{X}(t) = (x_1(t), \dots, x_n(t))^T$ represents the state of all nodes at time t , and the nonlinear function F is parameterized by the environment e .

Our task is zero-shot forecasting. We assume access to a set of historical trajectories collected from a number of *seen* environments, $E_{\text{seen}} \subset E$. The objective is to train a model that can accurately predict the future trajectory for a previously *unseen* environment $e_{\text{unseen}} \in E_{\text{unseen}}$, where the seen and unseen environment sets are disjoint ($E_{\text{seen}} \cap E_{\text{unseen}} = \emptyset$). Specifically, for a given trajectory, the forecasting task is defined as predicting a future window of states $\mathbf{X}_{t+1:t+N}$ given an observed historical window $\mathbf{X}_{t-H+1:t}$, where H is the look-back window size and N is the prediction horizon.

2.2 Conditional Flow Matching

Flow Matching is a powerful and recently developed generative modeling framework designed to learn a transformation from a simple prior distribution, p_0 , to a complex data distribution, p_1 [27, 28, 31]. This is achieved by training a parameterized, time-dependent vector field, $v_\xi(x, t)$, that learns to match a target velocity field guiding the transformation. Conditional Flow Matching (CFM) extends this concept by allowing the transformation to be dependent on a conditioning variable, c . The goal is thus to learn a map from p_0 to a conditional target distribution $p_1(x|c)$.

While various path definitions are possible, a common and effective approach is to use a straight-line path between samples from the source and target distributions [39]. Specifically, for a pair of samples $x_0 \sim p_0$ and $x_1 \sim p_1(\cdot|c)$, the probability path $p_t(x|x_0, x_1)$ is defined as a Gaussian bridge:

$$p_t(x|x_0, x_1) = \mathcal{N}(x | (1-t)x_0 + tx_1, \sigma^2), \quad (1)$$

where $t \in [0, 1]$ and σ^2 is a small variance. A key advantage of this formulation is that the corresponding target velocity field simplifies to a constant vector:

$$u_t(x|x_0, x_1) = x_1 - x_0. \quad (2)$$

This provides a direct and stable regression target for the conditional neural network $v_\xi(x, t, c)$. The network's weights ξ are optimized by minimizing the following loss function:

$$\mathcal{L}_{CFM}(\xi) = \mathbb{E}_{t, c, x_0, x_1} \left[\|v_\xi((1-t)x_0 + tx_1, t, c) - (x_1 - x_0)\|^2 \right], \quad (3)$$

where the expectation is taken over time $t \sim \mathcal{U}(0, 1)$, the conditioning variable c , prior samples $x_0 \sim p_0$, and target samples $x_1 \sim p_1(x|c)$. To further improve efficiency, modern implementations often pair samples x_0 and x_1 using mini-batch optimal transport (OT) plans, resulting in shorter and more direct flows [39].

3 Methodology

To address the challenge of zero-shot forecasting, we introduce Forecasting of Network Dynamics through Weight Flow Matching (FNFM), a novel generative framework. The core paradigm of FNFM shifts from directly predicting dynamic trajectories to generating the *weights* of a specialized forecaster model tailored to any given environmental condition. As illustrated in Figure 2, our methodology accomplishes this through a three-fold pipeline. FNFM first collects weights from expert models, then uses a variational autoencoder to map them into a latent space, and finally trains a

conditional flow matching model to generate latent vectors for the zero-shot synthesis of new models.

3.1 Collecting Expert Model Weights

We conceptualize the optimized weights of an expert forecaster trained on a single environment as a high-dimensional vector that captures the essence of that environment's unique network dynamics. Our FNFM framework is designed to learn the joint distribution of these weights and their corresponding environmental coefficients in a data-driven manner. Therefore, the foundational step of our methodology is to construct a dataset of these expert weights for all *seen* environments.

To achieve this, for each seen environment $e \in E_{\text{seen}}$ and its associated trajectory data $X^{(e)}$, we train a dedicated forecaster to parameterize the dynamical function $F_\theta(\cdot, A)$. While our framework is agnostic to the specific forecaster architecture, we employ a Spatio-Temporal Graph Convolutional Network (STGCN) [45] in our implementation. The weights $\theta^{(e)}$ for each expert are optimized by minimizing a multi-step forecasting loss over sliding windows of length H with a prediction horizon of N :

$$\theta^{(e)} = \arg \min_{\theta} \sum_{t=H}^{T-H-N} \|F_\theta(X_{t-H+1:t}^{(e)}, A) - X_{t+1:t+N}^{(e)}\|^2, \quad (4)$$

where $\theta^{(e)}$ denotes the resulting weights of the expert model tailored to environment e .

Each expert model is trained to converge on its specific environmental data using the Adam optimizer [19]. The final collection of optimized weights, $\{\theta^{(e)} \mid e \in E_{\text{seen}}\}$, serves as the target data for the subsequent generative learning stages of our framework.

3.2 Weight Sequence Tokenizer

We treat the weights of a neural network as a novel data modality. Fundamentally, these weights constitute a complex, structured representation, not merely a flat vector of numbers. To make this data compatible with powerful sequence-based models (like Transformers) while preserving the network's inherent architectural inductive biases, we introduce a tokenization scheme guided by the data flow through the network's computational units.

Our process operates on the fundamental building blocks of most neural networks: convolutional and linear layers [20, 22, 23]. For each expert model's weights, we decompose them layer by layer into a sequence of meaningful tokens.

Convolutional Layers. For a convolutional layer ℓ with kernel tensor $\Omega_\ell \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times h \times w}$, we form one token for each output channel. Each token aggregates all the weights responsible for producing that single output channel's feature map. Concretely, the token for the o -th output channel is:

$$\mathbf{w}_o^{(\ell)} = \text{flatten}(\Omega_{\ell, o, :, :}) \in \mathbb{R}^{C_{\text{in}} \cdot h \cdot w}, \quad \text{for } o = 1, \dots, C_{\text{out}}.$$

Linear Layers. Similarly, for a linear layer ℓ with weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$ and bias vector $\mathbf{b}_\ell \in \mathbb{R}^{D_{\text{out}}}$, we define one token per output unit (neuron). This token includes all incoming weights and the bias for that unit:

$$\mathbf{w}_o^{(\ell)} = [\mathbf{W}_{\ell, o, :}; \mathbf{b}_{\ell, o}] \in \mathbb{R}^{D_{\text{in}}+1}, \quad \text{for } o = 1, \dots, D_{\text{out}}.$$

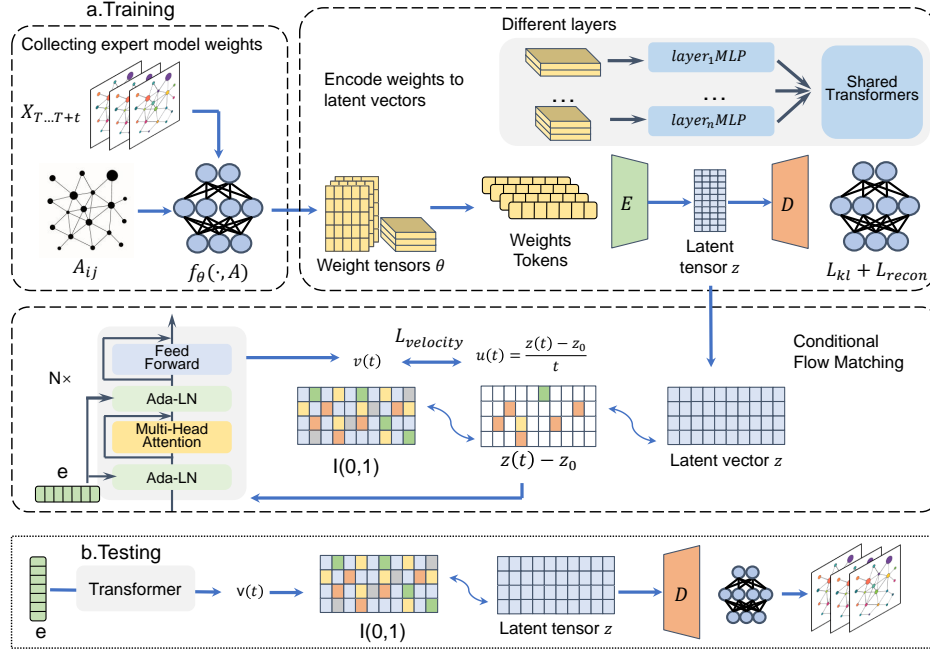


Figure 2: Overview of the Model Architecture. The framework comprises collecting expert model weights, tokenizing and encoding the model weight to latent vectors and conditional flow matching, working synergistically for zero-shot weight generation and dynamics forecasting.

This procedure losslessly transforms the entire weights θ of a L -layer neural network into an ordered sequence of tokens:

$$\{\mathbf{w}_1^{(1)}, \dots, \mathbf{w}_{C_{out}}^{(1)}, \dots, \mathbf{w}_1^{(L)}, \dots, \mathbf{w}_{D_{out}}^{(L)}\}.$$

Each token represents a self-contained computational unit, and the sequence preserves the layer-wise structure of the original model. This tokenized sequence serves as the direct input for our subsequent generative modeling stage.

3.3 Weight Variational Autoencoder

To facilitate stable and effective generative learning, we first compress the high-dimensional weight token sequence into a smooth and compact low-dimensional latent space [3]. We achieve this using a purpose-built Variational Autoencoder (VAE) featuring a Transformer-based architecture.

3.3.1 Model Architecture. The VAE consists of an encoder E that maps a sequence of weight tokens to a latent vector z , and a decoder D that reconstructs the token sequence from that vector.

Layer-wise Token Embedding. The raw tokens $\{\mathbf{w}_o^{(\ell)}\}$ from different layers possess varying dimensionalities, which is incompatible with a standard Transformer. To handle this, we first employ a set of layer-wise projection networks (MLPs), f_ℓ , to map each raw token into a fixed-dimensional embedding space:

$$\mathbf{h}_o^{(\ell)} = f_\ell(\mathbf{w}_o^{(\ell)}) \in \mathbb{R}^{d_{model}}. \quad (5)$$

Transformer Encoder. The resulting uniform-sized embeddings are fed into a multi-block Transformer encoder. Each block applies multi-head self-attention followed by a position-wise feed-forward

network with residual connections and layer normalization:

$$\mathbf{A} = \text{Concat}(\text{head}_1, \dots, \text{head}_k) \mathbf{W}^O, \quad (6)$$

$$\text{where } \text{head}_i = \text{Attention}(\mathbf{H} \mathbf{W}_i^Q, \mathbf{H} \mathbf{W}_i^K, \mathbf{H} \mathbf{W}_i^V), \quad (7)$$

$$\mathbf{H}' = \text{LayerNorm}(\mathbf{H} + \mathbf{A}), \quad (8)$$

$$\mathbf{H}_{out} = \text{LayerNorm}(\mathbf{H}' + \text{FFN}(\mathbf{H}')). \quad (9)$$

The final representation of each token is then passed through two separate linear layers to parameterize the mean μ and log-variance $\log \sigma^2$ of the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{w})$. A latent vector \mathbf{z} is then sampled using the reparameterization trick.

Transformer Decoder. The decoder mirrors the encoder's architecture. It takes the latent vector \mathbf{z} as a global conditioning input and reconstructs the sequence of embeddings. Finally, a set of layer-wise output networks, g_ℓ , project the decoder's output embeddings from $\mathbb{R}^{d_{model}}$ back to their original, layer-specific token dimensions to produce the reconstructed weights $\hat{\mathbf{w}}$.

3.3.2 Training Objective. Let ϕ and ψ represent the learnable parameters of encoder E and decoder D . The entire VAE is trained end-to-end by maximizing the Evidence Lower Bound (ELBO) on the log-likelihood of the weights:

$$\mathcal{L}_{\text{ELBO}}(\mathbf{w}; \phi, \psi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{w})} [\log p_\psi(\mathbf{w}|\mathbf{z})] - \beta \cdot \text{KL} [q_\phi(\mathbf{z}|\mathbf{w}) \parallel p(\mathbf{z})]. \quad (10)$$

The objective consists of two key terms. The first is the reconstruction loss, which measures the fidelity between the original and reconstructed weights, implemented as the negative mean squared error. The second is the Kullback-Leibler (KL) regularizer, which encourages the learned latent distribution $q_\phi(\mathbf{z}|\mathbf{w})$ to align with

a simple prior $p(\mathbf{z})$, typically a standard Gaussian $\mathcal{N}(0, I)$. This regularization ensures that the latent space is smooth and well-structured, which is crucial for the subsequent generative process.

Upon convergence, the trained encoder E_ϕ provides a robust mapping from any set of high-dimensional expert weights \mathbf{w} to a compact latent representation \mathbf{z} . This collection of latent vectors, $\{\mathbf{z}^{(e)} \mid e \in E_{\text{seen}}\}$, forms the target data manifold for our conditional flow matching module.

3.4 Conditional Flow Matching

With the VAE encoder providing a mapping to a structured latent space, the final stage of our framework is to learn a conditional generative model within this space. The goal is to synthesize a novel latent vector \mathbf{z}_{new} that corresponds to a previously unseen environmental coefficient e_{new} . We achieve this by training and deploying a conditional vector field using the flow matching principles outlined in the preliminaries.

3.4.1 Training the Conditional Vector Field. We train a time-dependent conditional vector field, parameterized by a neural network $v_\xi(\mathbf{z}, t, e)$, to approximate the target velocity field $(\mathbf{z}^{(e)} - \mathbf{z}_0)$ defined in Equation 2. The network’s parameters ξ are optimized by minimizing the following objective:

$$\mathcal{L}_{CFM}(\xi) = \mathbb{E}_{t, e, \mathbf{z}_0, \mathbf{z}^{(e)}} \left[\left\| v_\xi \left((1-t)\mathbf{z}_0 + t\mathbf{z}^{(e)}, t, e \right) - (\mathbf{z}^{(e)} - \mathbf{z}_0) \right\|^2 \right], \quad (11)$$

where the expectation is over time $t \sim \mathcal{U}(0, 1)$, seen environments $e \sim E_{\text{seen}}$, prior samples $\mathbf{z}_0 \sim \mathcal{N}(0, I)$, and their corresponding target latent codes $\mathbf{z}^{(e)} = E_\phi(\mathbf{w}^{(e)})$.

The vector field v_ξ is implemented using a Transformer architecture. To inject the environmental information e effectively, we employ an Adaptive Layer Normalization (AdaLN) mechanism. Within each Transformer block, the input sequence H_n is modulated before the self-attention layer:

$$\text{AdaLN}(H_n, e) = \gamma(e) \odot \text{LayerNorm}(H_n) + \beta(e), \quad (12)$$

where the scale $\gamma(e)$ and shift $\beta(e)$ are vectors produced from the environmental coefficient e by small multi-layer perceptrons. This allows the network’s behavior to be dynamically controlled by the target environment.

3.4.2 Zero-Shot Weight Generation via Inference. At inference time, FNFM generates a specialized set of weights for any unseen environment e_{new} in a zero-shot fashion. This generation process is framed as solving an ordinary differential equation (ODE) initial value problem. Starting with a random sample $\mathbf{z}_0 \sim \mathcal{N}(0, I)$, we integrate the learned vector field v_ξ from $t = 0$ to $t = 1$:

$$\frac{d\mathbf{z}_t}{dt} = v_\xi(\mathbf{z}_t, t, e_{\text{new}}), \quad \text{with initial value } \mathbf{z}_0. \quad (13)$$

This ODE is solved numerically using a standard solver such as forward Euler. For N integration steps, the update rule is:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{1}{N} v_\xi(\mathbf{z}_k, \frac{k}{N}, e_{\text{new}}), \quad \text{for } k = 0, \dots, N-1. \quad (14)$$

The resulting vector at the final step, $\mathbf{z}_N \approx \mathbf{z}_1$, is the synthesized latent representation for the new environment. This vector is then passed through the pre-trained VAE decoder D to generate the final,

ready-to-use forecaster weights $\hat{\mathbf{w}}_{\text{new}} = D(\mathbf{z}_N)$. The full training and inference procedure is provided in Algorithm 1 in Appendix A.

4 Experiment

4.1 Evaluation Protocol

We evaluate our method on the task of forecasting networked dynamical systems under distribution shifts. We regard an **environment** as the combination of a specific network’s trajectory data and its associated dynamic coefficients. For each dataset, we partition the available environments into training, validation, and testing sets, ensuring no overlap. The core of our evaluation lies in the test set, which is further divided into two distinct regions to rigorously assess generalization:

- **In-Domain:** Test environments whose dynamic coefficients are *interpolated* from within the range of coefficients observed during training.
- **Out-of-Domain:** Test environments whose dynamic coefficients are *extrapolated* beyond the range of the training set coefficients. This presents a more challenging test of a model’s generalization capabilities.

4.2 Implementation Details

We report multi-step forecasting performance using Root Mean Squared Error (RMSE) computed on the non-standardized trajectories. Across all experiments, we set the historical look-back window to $H = 50$ and the prediction horizon to $N = 50$. All model hyperparameters, for both our method and the baselines, are tuned on a dedicated validation set of environments. To ensure robust results, all reported metrics are the average of 5 independent runs using different random seeds but identical environment splits.

4.3 Datasets

We evaluate FNFM on five datasets covering both synthetic and real-world network topologies with heterogeneous dynamics. The synthetic datasets include **Hill**, which is generated on Barabási–Albert networks, while the datasets on real-world topology consist of **Epidemic** (European road network), **Twitter** (social propagation), and **Collab** (scientific collaboration). Each dataset contains multiple environments defined by distinct dynamic coefficients, with strictly non-overlapping training, validation, and test splits. Full details on governing equations, simulations, and OOD settings are provided in Appendix C.

4.4 Baselines

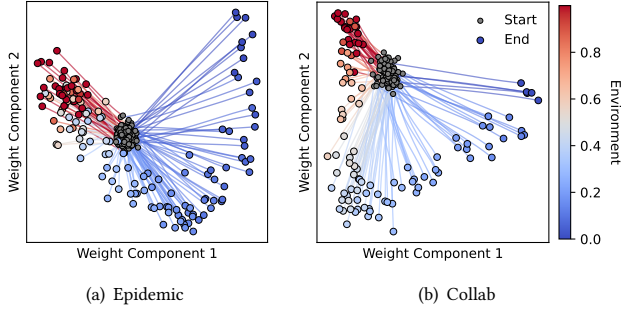
To assess the effectiveness of our approach, we compare FNFM against a comprehensive set of baselines ranging from standard forecasting models to advanced adaptive frameworks. The baselines include:

- Standard spatiotemporal models: **STGCN** [45] and **STEP** [37].
- Few-shot and generative approaches: **STGFSL** [30] and **GPD** [47].
- Adaptive and expert models: **Paragon** [38] and an oracle-like **One-per-Env STGCN**.

Please refer to Appendix D for descriptions of each baseline model.

Table 1: Average RMSE (\pm std from 5 runs) in various environments (split shown in the first row). Best in bold, underlined for suboptimal. STGFSL adopts a parameter-free meta learning strategy, so it has no additional parameters

Methods	Params	Hill		Epidemic		Twitter		Collab	
		In-domain	Out-domain	In-domain	Out-domain	In-domain	Out-domain	In-domain	Out-domain
STGCN[45]	13M	<u>14.4060\pm2.3449</u>	8.2359\pm0.8919	0.3797 \pm 0.0166	0.2437 \pm 0.0189	0.4402 \pm 0.0400	0.3156 \pm 0.0112	0.9153 \pm 0.0613	0.8945 \pm 0.0618
STEP [37]	11M	15.4436 \pm 0.6322	13.6226 \pm 0.5231	<u>0.0639\pm0.0021</u>	<u>0.0609\pm0.0042</u>	<u>0.0612\pm 0.0014</u>	<u>0.0593 \pm0.0013</u>	<u>0.0556\pm0.0033</u>	0.0303 \pm 0.0041
STGFSL[30]	-	53.2770 \pm 5.5370	38.7490 \pm 5.3160	0.1875 \pm 0.0005	0.3710 \pm 0.0008	0.3040 \pm 0.0010	0.4820 \pm 0.0010	0.0660 \pm 0.0007	0.0330 \pm 0.0012
Paragon [38]	11M	54.2730 \pm 1.8253	47.9000 \pm 1.7484	0.2367 \pm 0.0284	0.1333 \pm 0.0319	0.1333 \pm 0.0113	0.1584 \pm 0.0204	0.0755 \pm 0.1448	0.0326 \pm 0.0492
GPD [47]	12M	98.1748 \pm 0.0963	9.92475 \pm 1.315	0.1888 \pm 0.0002	0.0708 \pm 0.0021	0.1336 \pm 0.0004	0.0676 \pm 0.0020	0.0722 \pm 0.0001	<u>0.0280\pm0.0004</u>
Ours	11M	13.8942 \pm2.4240	<u>8.5595 \pm 0.4594</u>	0.0562 \pm0.0071	0.0561\pm0.0059	0.0579 \pm 0.0081	0.0512\pm0.0048	0.0475 \pm 0.0036	0.0244\pm0.0021
Percentage		3.55%	-3.93%	12.05%	7.89%	5.39%	13.66%	14.57%	12.86%
One-per-Env STGCN		11.2409	8.5127	0.0496	0.0538	0.0364	0.0326	0.0461	0.0229

**Figure 3: The process of FNFM generating weights for a predictive model under different environmental conditions.**

4.5 Main Results

Table 1 presents the comparative results on four datasets. Our proposed method, FNFM, consistently achieves state-of-the-art performance, outperforming all baselines across nearly every in-domain and out-of-domain scenario. Notably, FNFM’s accuracy is highly competitive with the One-per-Env STGCN, an oracle-like expert model trained with full access to data from the target environment.

FNFM’s effectiveness is attributable to its ability to explicitly learn the low-dimensional manifold of the expert model weights. While monolithic baselines seek a single compromise model and meta-learning requires target data to navigate this space, FNFM learns the global structure of the manifold itself. By using a VAE to identify this structure and a CFM to learn the direct map from any environmental coefficient to a point upon it, our framework can instantly generate a specialized, near-optimal model.

4.6 Explainability

To gain deeper insight into the internal workings of FNFM, we visualize its weight generation process on two datasets in Figure 3. We use principal component analysis to project the latent space learned by the VAE onto a two-dimensional plane. The visualization clearly reveals three key elements of our framework’s success.

First, the latent vectors of the expert models (the End points) are not scattered randomly; instead, they converge to form a smooth, well-structured, low-dimensional manifold. This indicates that the space of effective model weights possesses a strong intrinsic structure, which our VAE successfully captures.

Second, this manifold is meaningfully organized by the environmental coefficients, as illustrated by the color gradient. We observe a continuous color transition along the manifold’s structure, signifying that similar environments correspond to proximate locations in the latent space. This confirms that our framework has learned a semantic mapping from dynamic environments to model weights.

Finally, the trajectories connecting the start points (from the Gaussian prior) to the end points (the target weights) visualize the conditional flow matching process. These trajectories follow direct, nearly-straight paths from the simple prior to their target locations on the manifold. This reveals that our CFM model has learned a stable and efficient transport map, ensuring high-quality zero-shot generation.

In summary, this visualization provides compelling evidence for FNFM’s success: it learns not only a semantically organized manifold of expert models but also an effective conditional path to navigate it.

4.7 Case Study

To further investigate FNFM’s ability to generalize, we conduct a case study on the Collab dataset, which models an SIS-like information propagation process on a network of scientific collaborations (Figure 4a). This system exhibits a critical phenomenon known as a phase transition, where the long-term outcome of the dynamics is acutely sensitive to the environmental coefficient, which in this context represents the information’s stickiness [40].

As illustrated in Figure 4b, the system’s final propagation scale displays three distinct regimes based on the environmental coefficient. When the coefficient is below a critical threshold (the Declining region), activity eventually dies out. Conversely, above a higher threshold (the Active region), the information becomes endemic, reaching a high, stable level of activity. Between these extremes lies a highly non-linear Transition region, where small changes in the coefficient lead to dramatic shifts in the outcome.

For this experiment, we deliberately trained FNFM only on data from the two extreme regimes (Declining and Active), leaving the entire critical Transition region as a challenging, unseen test bed. The results demonstrate a remarkable generalization capability. Figure 4c shows that FNFM’s zero-shot predictions for the propagation scale in this unseen region align closely with the ground truth, indicating that our model successfully learned the underlying non-linear function governing the phase transition. Furthermore, Figure 4d displays two example trajectory forecasts, confirming

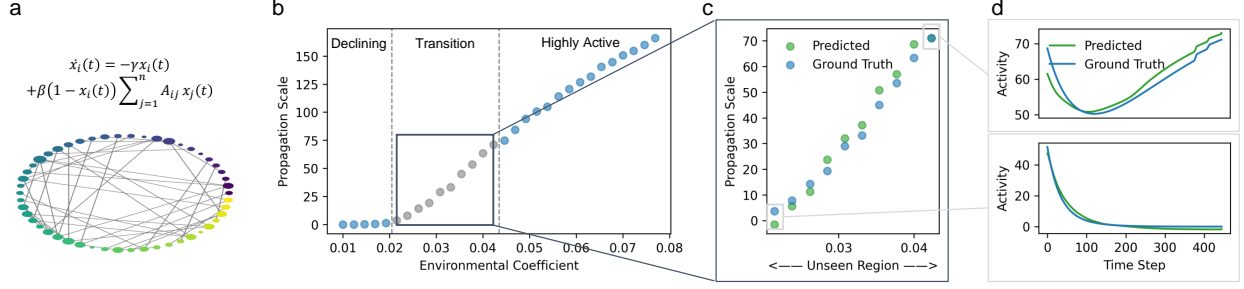


Figure 4: Case study on the Collab dataset. (a) Illustration of the network topology and governing equation. (b) The information propagation scale (network activity at the final time step) as a function of the environmental coefficient (popularity). (c) FNFM’s generalized prediction of the propagation scale within the phase transition region closely matches the ground truth. (d) FNFM’s predicted trajectories for two extreme scenarios: a declining case and an active case.

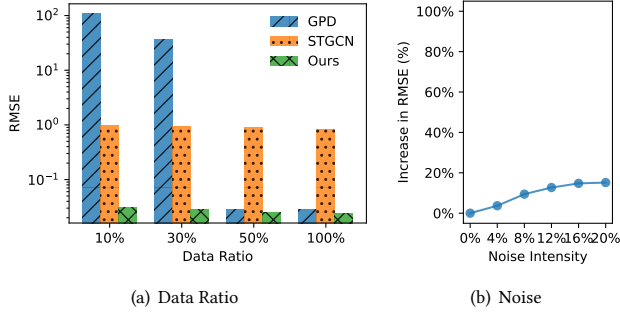


Figure 5: Robustness results on the Collab dataset.

that our generated models can accurately predict the full temporal evolution of the system’s network-wide activity.

This case study provides strong evidence that FNFM learns more than simple input-output mappings; it captures the fundamental principles of a complex system’s critical behavior. The ability to accurately interpolate within a phase transition showcases its potential as a powerful tool for reliable forecasts for systems with novel parameters near critical tipping points.

4.8 Robustness

We conduct two additional experiments on the Collab dataset to assess the robustness of FNFM under challenging conditions: limited data availability and noisy environmental coefficients.

Robustness to Limited Data. In this experiment, we evaluate model performance when trained on a reduced number of available environments (from 100% down to 10%). As shown in Figure 5a, FNFM maintains its superior performance and low RMSE even when trained with only 10% of the environments. In contrast, the competing generative model, GPD, suffers a catastrophic performance degradation under data scarcity, indicating its heavy reliance on a large number of training examples. This highlights FNFM’s excellent data efficiency, suggesting that our framework can effectively learn the underlying manifold of expert weights from a very limited sample of environments.

Robustness to Noisy Coefficients. This experiment tests the model’s stability when the provided environmental coefficients at

inference time are inaccurate. We add zero-mean Gaussian noise (standard deviation is setting to the coefficient’s total range) to the true coefficients, with the noise intensity varying from 0% to 20%. Figure 5b shows that the RMSE of FNFM increases gracefully and smoothly as the noise intensity grows, with less than a 16% increase in error even at 20% noise. This smooth degradation, rather than a sudden breakdown, provides strong evidence that FNFM has learned a continuous and well-behaved mapping from the coefficient space to the latent space of model weights. This property is crucial for practical applications, as it ensures that small estimation errors in the environmental coefficients will only lead to small and predictable errors in the final forecast.

4.9 Ablation Study

To validate the effectiveness of our key design choices, we conduct an ablation study on the Hill and Collab datasets, with results shown in Table 2, including a backbone comparison with DDPM.

Weight Sequence Tokenizer This experiment assesses the importance of our structure-preserving tokenizer. In the “w/o Tokenizer” variant, we simply flatten all model weights into a single vector and then reshape it into a token sequence, disrupting the network’s architectural inductive biases. The results show that this naive approach leads to a notable performance degradation on the Collab dataset, particularly in the out-of-domain split. This validates our hypothesis that preserving the computational structure of the weights is crucial for the generative model to learn a meaningful and generalizable representation.

Environmental Coefficients This experiment evaluates the necessity of the conditional generation mechanism. In the “w/o Condition” variant, the CFM model is trained unconditionally to generate an “average” expert model. The results show a severe drop in performance across all datasets and splits. This confirms that the environmental coefficient is the essential guiding signal for synthesizing the correct, specialized model weights. Without this conditioning, the model fails to adapt to the specific dynamics of any given environment, underscoring the critical role of our conditional framework.

Necessity of VAE To justify the use of a Variational Autoencoder (VAE) for weight compression, we tested a variant “w/o VAE” where flattened weights are fed directly into the CFM model. As shown in

Table 2: Ablation studies on Hill and Collab datasets. ‘w/o’ stands for ‘without’. Lower RMSE indicates better performance.

Variant	Hill (RMSE)		Collab (RMSE)	
	In	Out	In	Out
w/o VAE	118.9763	142.2945	0.0495	0.0340
w/o Tokenizer	17.3381	11.1205	0.05134	0.0359
w/o Condition	27.6437	17.5250	0.0670	0.0306
Ours (CFM)	13.8942	8.5595	0.0475	0.0244
DDPM (50 steps)	18.5735	11.4027	0.1021	0.0529
CFM (50 steps)	14.9386	8.7411	0.0483	0.0247

Table 2, removing the VAE causes severe degradation, particularly on the Hill dataset (RMSE increases from 13.89 to 118.97). This confirms that raw weight spaces are too high-dimensional and sparse for efficient direct generation. Compressing weights into a compact, regularized latent space is essential for effective flow matching and stable training.

CFM vs. Diffusion (DDPM) We further compare our chosen Continuous Flow Matching (CFM) backbone against a standard Denoising Diffusion Probabilistic Model (DDPM). We replaced CFM with DDPM while keeping other components unchanged. Both models were evaluated using a fixed budget of 50 sampling steps to reflect a computationally efficient inference scenario. As shown in Table 2, CFM consistently achieves lower RMSE across both datasets and splits under this constraint. Theoretically, CFM learns straighter optimal transport trajectories in the latent space compared to the stochastic paths of diffusion models. This property enables faster convergence and more accurate weight generation even with fewer sampling steps (50 steps).

5 Related work

5.1 Modeling of Network Dynamics

Data-driven modeling of complex network dynamics, particularly with Graph Neural Networks (GNNs), has become a prominent research direction. Foundational models like STGCN [45] and Graph WaveNet [43] established effective frameworks for spatio-temporal forecasting by integrating graph convolutions with temporal modeling. Subsequent research has advanced this field by incorporating more sophisticated mechanisms, such as using Neural Ordinary Differential Equations (ODEs) to capture continuous-time dynamics [48], or designing specific encoders to handle dynamic topologies and generalize across different environments [15, 16].

However, a fundamental challenge for these models is generalizing to unseen dynamic regimes. To address this, one line of work focuses on building large-scale, "one-for-all" foundation models that train on diverse data [24, 37]. While powerful, they often sacrifice the specialized accuracy required for specific dynamic conditions. Another prominent approach leverages meta-learning to quickly adapt a base model to new environments [6, 30]. Despite their adaptability, these methods typically require at least a small

amount of trajectory data from the target environment for fine-tuning, precluding their use in true zero-shot scenarios. Our work targets this critical gap.

5.2 Generative Models for Network Weights

Generating neural network weights is an emerging paradigm with significant potential for generalization [42]. This area has evolved along several fronts. One initial line of work focused on generating weights to accelerate or improve the training process itself, effectively replacing hand-crafted initializations [10, 36].

More recent studies, closer to our own, leverage conditional generative models like diffusion to produce weights tailored for generalization. For instance, Yuan et al. [47] use an urban knowledge graph as a prompt to generate spatio-temporal models for unseen cities. Others have integrated weight generation into the meta-learning loop, replacing gradient-based inner-loop [49], or proposed controllable frameworks like Paragon [38], which employs parameter diffusion for test-time adaptation. While these methods represent significant progress, their zero-shot performance is often limited, as many still necessitate post-generation finetuning.

A potential reason for this limitation lies in the representation of the weights themselves. Most existing methods treat the weights as a simple flat vector, disrupting the network’s inherent architectural inductive biases and making the distribution harder for a generative model to learn. To address this Deep Weight Flow [12] applies re-basing techniques within a Flow Matching framework to account for permutation symmetries, generating high-accuracy weights that do not require fine-tuning. In contrast, our work introduces two key innovations: 1) We employ a novel weight sequence tokenizer that preserves the computational structure of the network, providing a more meaningful representation for the generative model. 2) By synergistically combining a VAE and conditional flow matching, we directly learn a smooth manifold of expert weights, enabling truly zero-shot, single-pass generation of high-performance models without any subsequent tuning.

6 Conclusion

This work introduced FNFM, a novel generative framework that successfully addresses the critical challenge of zero-shot prediction for network dynamics. Recognizing that dynamics are highly sensitive to their governing coefficients, we proposed a paradigm shift: from directly predicting trajectories to generating the complete weights of a specialized forecaster model itself. Our methodology operationalizes this concept through a synergistic combination of a variational autoencoder and a conditional flow matching model. The VAE first learns a compact and smooth latent manifold of expert model weights, after which the CFM framework learns to map any environmental coefficient to a specific location on this manifold. This mechanism enables the instantaneous, single-pass generation of tailored models for unseen environments, demonstrating robust generalization without any need for finetuning.

A current limitation of FNFM is that our current framework assumes a static network topology; extending FNFM to handle dynamic graphs where nodes and edges evolve over time is a critical next step.

Acknowledgments

This work was supported by the Science and Technology Innovation Program of Xiongan New Area under Grant No. 2025XAGG0041; and the National Natural Science Foundation of China under Grant No. U23B2030.

References

- [1] Linda JS Allen. 1994. Some discrete-time SI, SIR, and SIS epidemic models. *Mathematical biosciences* 124, 1 (1994), 83–105.
- [2] Alexandre Bovet and Hernán A Makse. 2019. Influence of fake news in Twitter during the 2016 US presidential election. *Nature communications* 10, 1 (2019), 7.
- [3] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. 2023. Flow matching in latent space. *arXiv preprint arXiv:2307.08698* (2023).
- [4] Jingtao Ding, Yu Zheng, Huandong Wang, Carlo Vittorio Cannistraci, Jianxi Gao, Yong Li, and Chuan Shi. 2025. Artificial Intelligence for Complex Network: Potential, Methodology and Application. In *Companion Proceedings of the ACM on Web Conference 2025* (Sydney NSW, Australia) (WWW '25). Association for Computing Machinery, New York, NY, USA, 5–8. doi:10.1145/3701716.3715857
- [5] Neil M Ferguson, Daniel Laydon, Gemma Nedjati-Gilani, Natsuko Imai, Kylie Ainslie, Marc Baguelin, Sangeeta Bhatia, Adhiratha Boonyasiri, Zulma Cucunubá, Gina Cuomo-Dannenburg, et al. 2020. *Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand*. Vol. 16. Imperial College London London.
- [6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (ICML '17). JMLR.org, Sydney, NSW, Australia, 1126–1135.
- [7] Richard FitzHugh. 1961. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal* 1, 6 (1961), 445–466. doi:10.1016/S0006-3495(61)86902-6
- [8] Jay W Forrester. 1970. Urban dynamics. *IMR; Industrial Management Review (pre-1986)* 11, 3 (1970), 67.
- [9] Jesús Gómez-Gardeñes, David Soriano-Panos, and Alex Arenas. 2018. Critical regimes driven by recurrent mobility patterns of reaction–diffusion processes in networks. *Nature Physics* 14, 4 (2018), 391–395.
- [10] Yifan Gong, Zheng Zhan, Yanyu Li, Yerlan Idelbayev, Andrey Zharkov, Kfir Aberman, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. 2024. Efficient Training with Denoised Neural Weights. In *European Conference on Computer Vision*. 18–34.
- [11] Adam Goodge, Wee Siong Ng, Bryan Hooi, and See Kiong Ng. 2025. Spatio-temporal foundation models: Vision, challenges, and opportunities. *arXiv preprint arXiv:2501.09045* (2025).
- [12] Saumya Gupta, Scott Biggs, Moritz Laber, Zohair Shafi, Robin Walters, and Ayan Paul. 2026. DeepWeightFlow: Re-Basined Flow Matching for Generating Neural Network Weights. arXiv:2601.05052 [cs.LG] <https://arxiv.org/abs/2601.05052>
- [13] M Elizabeth Halloran, Neil M Ferguson, Stephen Eubank, Ira M Longini Jr, Derek AT Cummings, Bryan Lewis, Shufu Xu, Christophe Fraser, Anil Vullikanti, Timothy C Germann, et al. 2008. Modeling targeted layered containment of an influenza pandemic in the United States. *Proceedings of the National Academy of Sciences* 105, 12 (2008), 4639–4644.
- [14] Chittaranjan Hens, Uzi Harush, Simi Haber, Reuven Cohen, and Baruch Barzel. 2019. Spatiotemporal Signal Propagation in Complex Networks. *Nature Physics* 15, 4 (April 2019), 403–412. doi:10.1038/s41567-018-0409-0
- [15] Zijie Huang, Yizhou Sun, and Wei Wang. 2021. Coupled graph ode for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 705–715.
- [16] Zijie Huang, Yizhou Sun, and Wei Wang. 2023. Generalizing graph ode for learning complex system dynamics across environments. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 798–809.
- [17] Iacopo Iacopini, Márton Karsai, and Alain Barrat. 2024. The temporal dynamics of group interactions in higher-order social networks. *Nature Communications* 15, 1 (2024), 7391.
- [18] Marko Jusup, Petter Holme, Kiyoshi Kanazawa, Misako Takayasu, Ivan Romić, Zhen Wang, Sunčana Geček, Tomislav Lipić, Boris Podobnik, Lin Wang, et al. 2022. Social physics. *Physics Reports* 948 (2022), 1–148.
- [19] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [20] Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J. Burghouts, Efstratios Gavves, Cees G. M. Snoek, and David W. Zhang. 2024. Graph Neural Networks for Learning Equivariant Representations of Neural Networks. arXiv:2403.12143 [cs] doi:10.48550/arXiv.2403.12143
- [21] David Lazer, Alex Pentland, Lada Adamic, Sinan Aral, Albert-László Barabási, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. 2009. Computational social science. *Science* 323, 5915 (2009), 721–723.
- [22] Ruikun Li, Jiazhen Liu, Huandong Wang, Qingmin Liao, and Yong Li. 2025. WeightFlow: Learning Stochastic Dynamics via Evolving Weight of Neural Network. *arXiv preprint arXiv:2508.00451* (2025).
- [23] Ruikun Li, Huandong Wang, Jingtao Ding, Yuan Yuan, Qingmin Liao, and Yong Li. 2025. Predicting Dynamical Systems across Environments via Diffusive Model Weight Generation. *arXiv preprint arXiv:2505.13919* (2025).
- [24] Ruikun Li, Huandong Wang, Jinghua Piao, Qingmin Liao, and Yong Li. 2024. Predicting long-term dynamics of complex networks via identifying skeleton in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1655–1666.
- [25] Zhonghang Li, Long Xia, Lei Shi, Yong Xu, Dawei Yin, and Chao Huang. 2024. Opacity: Open spatio-temporal foundation models for traffic prediction. *arXiv preprint arXiv:2408.10269* (2024).
- [26] Yuxuan Liang, Haomin Wen, Yutong Xia, Ming Jin, Bin Yang, Flora Salim, Qingsong Wen, Shirui Pan, and Gao Cong. 2025. Foundation models for spatio-temporal data science: A tutorial and survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 6063–6073.
- [27] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747* (2022).
- [28] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow Matching for Generative Modeling. arXiv:2210.02747 [cs] doi:10.48550/arXiv.2210.02747
- [29] Jiazhen Liu, Shengda Huang, Nathaniel M Aden, Neil F Johnson, and Chaoming Song. 2023. Emergence of polarization in coevolving networks. *Physical Review Letters* 130, 3 (2023), 037401.
- [30] Bin Lu, Xiaoying Gan, Weinan Zhang, Huaxiu Yao, Luoyi Fu, and Xinbing Wang. 2022. Spatio-Temporal Graph Few-Shot Learning with Cross-City Knowledge Transfer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1162–1172. doi:10.1145/3534678.3539281
- [31] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vandenberg, and Saining Xie. 2024. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*. Springer, 23–40.
- [32] Fanhui Meng, Jiarong Xie, Jiachen Sun, Cong Xu, Yutian Zeng, Xiangrong Wang, Tao Jia, Shuhong Huang, Youjin Deng, and Yanqing Hu. 2025. Spreading dynamics of information on online social networks. *Proceedings of the National Academy of Sciences* 122, 4 (2025), e2410227122.
- [33] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1720–1730.
- [34] Zheyi Pan, Wentao Zhang, Yuxuan Liang, Weinan Zhang, Yong Yu, Junbo Zhang, and Yu Zheng. 2020. Spatio-temporal meta learning for urban traffic prediction. *IEEE Transactions on Knowledge and Data Engineering* 34, 3 (2020), 1462–1476.
- [35] Huiling Qin, Songyu Ke, Xiaodu Yang, Haoran Xu, Xianyu Zhan, and Yu Zheng. 2021. Robust spatio-temporal purchase prediction via deep meta learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4312–4319.
- [36] Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. 2022. Hyper-representations as generative models: Sampling unseen neural network weights. *Advances in Neural Information Processing Systems* 35 (2022), 27906–27920.
- [37] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-Training Enhanced Spatial-Temporal Graph Neural Network for Multivariate Time Series Forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1567–1577. doi:10.1145/3534678.3539396
- [38] Chenglei Shen, Jiahao Zhao, Xiao Zhang, Weijie Yu, Ming He, and Jianping Fan. 2024. Generating Model Parameters for Controlling: Parameter Diffusion for Controllable Multi-Task Recommendation. *arXiv preprint arXiv:2410.10639* (2024).
- [39] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. 2024. Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport. *Transactions on Machine Learning Research* (2024).
- [40] Alessandro Vespignani. 2012. Modelling dynamical processes in complex socio-technical systems. *Nature physics* 8, 1 (2012), 32–39.
- [41] Huandong Wang, Huan Yan, Can Rong, Yuan Yuan, Fenyu Jiang, Zhenyu Han, Hongjie Sui, Depeng Jin, and Yong Li. 2024. Multi-scale Simulation of Complex Systems: A Perspective of Integrating Knowledge and Data. *ACM Comput. Surv.* 56, 12, Article 307 (Oct. 2024), 38 pages. doi:10.1145/3654662
- [42] Kai Wang, Dongwen Tang, Boya Zeng, Yida Yin, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. 2024. Neural network diffusion. *arXiv preprint arXiv:2402.13144* (2024).
- [43] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*.

- International Joint Conferences on Artificial Intelligence Organization, 1907–1913. doi:10.24963/ijcai.2019/264
- [44] Zhouzheng Xu, Yuxing Wu, Hang Zhou, Chaofan Fan, Bingyi Li, Kaiyue Liu, Yaqin Ye, Shunping Zhou, and Shengwen Li. 2025. Spatio-Temporal Meta-learning for Trajectory Representation Learning. *Knowledge-Based Systems* (2025), 114141.
- [45] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 3634–3640. doi:10.24963/ijcai.2018/505
- [46] Yuan Yuan, Chonghua Han, Jingtao Ding, Depeng Jin, and Yong Li. 2024. Urbandit: A foundation model for open-world urban spatio-temporal learning. *arXiv preprint arXiv:2411.12164* (2024).
- [47] Yuan Yuan, Chenyang Shao, Jingtao Ding, Depeng Jin, and Yong Li. 2024. Spatio-Temporal Few-Shot Learning via Diffusive Neural Network Generation. In *The Twelfth International Conference on Learning Representations*.
- [48] Chengxi Zang and Fei Wang. 2020. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 892–902.
- [49] Baoquan Zhang, Chuyao Luo, Demin Yu, Xutao Li, Huiwei Lin, Yunming Ye, and Bowen Zhang. 2024. Metadiff: Meta-learning with conditional diffusion for few-shot learning. In *Proceedings of the AAAI conference on artificial intelligence*, 16687–16695.
- [50] Muhua Zheng, Linyuan Lü, and Ming Zhao. 2013. Spreading in online social networks: The role of social reinforcement. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 88, 1 (2013), 012818.

A Algorithmic Details

Algorithm 1 CFM for Conditional Latent Vector Generation

Require: Pre-computed latent vectors and conditions $\mathcal{Z} = \{(z^{(e)}, e)\}_{e \in E_{seen}}$

Ensure: Trained CFM network v_ξ

// Stage 1: Training the Conditional Vector Field

procedure TRAINCFM(\mathcal{Z})

Initialize CFM network parameters ξ .

for each training step **do**

Sample batch $\{(z_1^{(i)}, e^{(i)})\}_{i=1}^B$ from \mathcal{Z} .

Sample priors $z_0^{(i)} \sim \mathcal{N}(0, I)$ and times $t^{(i)} \sim \mathcal{U}(0, 1)$.

$z_t^{(i)} \leftarrow (1 - t^{(i)})z_0^{(i)} + t^{(i)}z_1^{(i)}$ \triangleright Construct path points

$u^{(i)} \leftarrow z_1^{(i)} - z_0^{(i)}$ \triangleright Define target velocities

$\mathcal{L}_{CFM} \leftarrow \frac{1}{B} \sum_{i=1}^B \|v_\xi(z_t^{(i)}, t^{(i)}, e^{(i)}) - u^{(i)}\|^2$ \triangleright Loss

Update ξ by descending the gradient of \mathcal{L}_{CFM} .

end for

end procedure

// Stage 2: Generating Latent Vectors via Inference

procedure GENERATELATENTVECTOR(v_ξ, e^{new}, N)

Sample prior $z_0 \sim \mathcal{N}(0, I)$.

for $k = 0, \dots, N - 1$ **do**

$z_{k+1} \leftarrow z_k + \frac{1}{N} \cdot v_\xi(z_k, k/N, e^{new})$ \triangleright Forward Euler step

end for

return z_N \triangleright Latent vector for new environment

end procedure

B Additional Experiments on FitzHugh-Nagumo Dynamics

Following the reviewers' suggestion, we additionally evaluate our method on the FitzHugh-Nagumo (FHN) dynamics, a classical nonlinear system characterized by high-frequency oscillations and

fast-slow interactions. Compared to the SIS and Hill dynamics considered in the main paper, FHN represents a different dynamical regime and is therefore reported here as a supplementary experiment. We empirically observe that STGCN may exhibit degraded predictive performance on FHN, possibly due to its limited model capacity and temporal inductive bias when modeling high-frequency oscillatory dynamics. To ensure a fair comparison across methods, we adopt Graph WaveNet[43] as the backbone model for all approaches in this experiment.

The results are summarized in Table 3. This experiment is intended to serve as a supplementary analysis and does not alter the main conclusions drawn from the SIS and Hill dynamics. The results indicate that our method achieves competitive performance relative to existing baselines under this additional dynamical setting.

Table 3: RMSE on the FHN dynamics using Graph WaveNet as the backbone model.

Method	In-domain	Out-domain
STGCN	0.0578	0.0909
STEP	0.0513	0.0475
STGFSL	0.1520	0.1160
GPD	0.5726	0.8553
Ours	0.0180	0.0193
Per-Env	0.0112	0.0153

C Dataset Details

C.1 Network Dynamics

Hill dynamics [14] describes regulatory interactions in sub-cellular networks, with the following governing equations:

$$\frac{dx_i}{dt} = -B_i x_i^a + \sum_{j=1}^N A_{ij} \frac{x_j^h}{1 + x_j^h}, \quad (15)$$

where $x_i(t)$ is the abundance of protein i , A_{ij} is the network topology, and the exponents a and h control the self-dynamics and regulatory interaction, respectively.

SIS dynamics [1, 40] models an epidemic process on a network where nodes can be infected by their neighbors and recover to a susceptible state. The governing equation for the infection probability x_i of each node i is:

$$\frac{dx_i}{dt} = -\gamma x_i + \beta(1 - x_i) \sum_{j=1}^N A_{ij} x_j, \quad (16)$$

where x_i is the state of node i , A_{ij} is the adjacency matrix representing the network topology, γ is the recovery rate, and β is the infection rate per contact.

FitzHugh-Nagumo (FHN) dynamics [7, 24] is a simplified model of neuronal excitable systems. In this networked version, each node i is described by a fast voltage variable $x_{i,1}$ and a slow recovery variable $x_{i,2}$. Based on the implementation, the governing equations are:

$$\begin{cases} \frac{dx_{i,1}}{dt} = x_{i,1} - x_{i,1}^3 - x_{i,2} + \frac{c}{k_i} \sum_j A_{ij} (x_{j,1} - x_{i,1}) \\ \frac{dx_{i,2}}{dt} = e + f x_{i,1} - \gamma x_{i,2} \end{cases}$$

Table 4: Simulation settings of each networked system.

	Hill (BA)	Epidemic	Twitter	Collab	FHN (BA)
Dynamics	Hill	SIS	SIS	SIS	FHN
Node Num	300	1,174	761	1,511	300
Edge Num	1,475	1,417	1,029	4,273	596
Avg Degree	9.83	2.41	2.70	5.66	3.97
Time Unit (s)	0.04	0.1	0.1	0.02	0.05
Time Length (s)	600.0	50.0	50.0	10.0	300.0
Environments	$a \in [0.5, 0.6, 10], h \in [0.3, 2.0, 10], B \in [0.2, 0.9, 4] \quad \beta \in [0.02, 0.08, 11], \gamma \in [0.01, 0.12, 11] \quad \beta \in [0.02, 0.04, 11], \gamma \in [0.01, 0.12, 11] \quad \beta = 0.02, \gamma \in [0.2, 2.0, 40] \quad e \in [0.0, 0.6, 20], f \in [0.0, 1.0, 20]$				

where A_{ij} is the adjacency matrix and $k_i = \sum_j A_{ij}$ is the **degree** of node i . The parameter c represents the **coupling weight**, while e and f are parameters controlling the system's **limit cycle behavior**. The constant γ denotes the **decay rate** of the inhibitor variable.

C.2 Simulation and Data Splitting

We employ Euler's method to numerically solve the dynamics of the aforementioned systems for each topology, generating evolutionary trajectories as the datasets. To create challenging generalization tasks, we vary the key dynamic coefficients to form distinct training and testing environments.

For the **Hill** dataset, the out-of-domain (OOD) environments are created by sampling coefficients from the boundaries of the training ranges; specifically, setting parameter $a = 0.6$ (from a training range of $[0.5, 0.6]$) and sampling parameter h from the sub-range $[1.2580, 2.0000]$ (from a full training range of $[0.33, 2.00]$).

For the **Epidemic** and **Twitter** datasets, the parameter β is varied across its full range while the coefficient γ is split. For Epidemic, the training range for γ is $[0.0200, 0.0330]$, while the OOD test set extrapolates to $[0.0360, 0.0390]$. For Twitter, the training range for γ is $[0.0200, 0.0740]$, with its OOD test set in the extrapolated range of $[0.0740, 0.0800]$.

For the **Collab** dataset, the parameter β is fixed at 0.02, while the training environments use a γ range of $[0.2000, 0.4264]$. The OOD test set for this dataset involves a distributional shift to a completely separate, non-overlapping range of $[0.4728, 0.9302]$.

For the **FHN** dataset, the training and OOD environments are partitioned based on the relationship between parameters e and f . The training and in-distribution test sets are sampled from the regions satisfying $|f - e| > 0.2$ (specifically $f > e + 0.2$ or $f < e - 0.2$) using a 7 : 3 split. To evaluate generalization, the OOD test set is constructed from the complementary "gap" region where $|f - e| \leq 0.2$, representing a dynamical transition zone excluded from the training phase.

D Baseline Descriptions

We compare our proposed method against the following baselines:

- **STGCN** [45]: A deep learning model that combines graph convolutions for spatial dependencies with gated temporal convolutions for temporal dynamics, jointly modeling graph-structured time series.
- **STEP** [37]: A novel framework that enhances spatiotemporal graph neural networks by using a pretraining model to learn temporal patterns and generate segment-level representations.

- **STGFSL** [30]: A model-agnostic, spatiotemporal graph few-shot learning framework designed for scenarios with limited data.
- **GPD** [47]: A generative pretraining framework that pretrains a diffusion model to generate customized parameters for spatiotemporal prediction networks.
- **Paragon** [38]: A controllable learning approach that utilizes parameter diffusion to adapt models to varying task requirements. It employs a diffusion-based generator to produce optimized model parameters in a test-time adaptation manner, eliminating the need for retraining when objectives change.
- **One-per-Env STGCN** [45]: A set of expert STGCN models trained specifically for each environment, serving as a reference for the upper bound of performance.

E Software and Hardware Environment

We implement FNFM in PyTorch and employ the open-source available implementations with default parameters for baselines. All experiments were conducted on the NVIDIA GeForce RTX 2080Ti GPU. For all datasets and baselines, we set the batch size to 64 and trained for 500 epochs with a learning rate of 0.0001.

F Model Configuration

Weight Variational Autoencoder. Our Weight Variational Autoencoder (VAE) is built upon a Transformer architecture designed to process the sequence of model weights. The VAE's internal model dimension (d_{model}) is set to 128. The Transformer architecture consists of 2 layers, each equipped with 8 attention heads. The encoder network compresses the input weight token sequence into a 32-dimensional latent space. For training, we used the Adam optimizer with an initial learning rate of $1e-4$ and a weight decay of $3e-9$, managed by a OneCycleLR scheduler. The batch size was set to 32, and the Kullback-Leibler (KL) divergence term in the ELBO loss was weighted by a β factor of $1e-6$.

Conditional Flow Matching Model. The conditional vector field (v_ξ) for the Flow Matching process is also implemented using a Transformer-based architecture. This network operates on the 32-dimensional latent space, taking a latent vector as input and outputting a velocity vector of the same dimension. The architecture is composed of 4 Transformer layers, each with 2 attention heads. A dropout rate of 0.1 is applied during training for regularization. The model is conditioned on external environmental information, which is provided through dedicated knowledge graph and time embeddings.