EvolveODE: Emulating Underlying Dynamics through Sorting and Coupling

Anonymous authors
Paper under double-blind review

Abstract

Observational-level data arises from processes beyond the scope of direct observation. These processes naturally couple across the scales they natively occupy. In real-world settings, they may include global weather subsystems and time-delayed traffic flows, among other such systems. Through self-attention, Graph Neural Networks (GNNs) offer a natural way to untangle these confounding factors, namely by leveraging self-organization to sort variables that strongly interact with one another, forming a sub-process. We propose EvolveODE, a multiscale GNN that solves attraction-repulsion equations to identify groupings, thereby learning the coupling between the physical processes underlying observation-level phenomena. EvolveODE then emulates these relationships with a dynamical system. We show that EvolveODE outperforms the current state-of-the-art baselines across various tasks and settings. Code is available at `https://anonymous.4open.science/r/EvolveODE/`.

1 Introduction

Meaningful inference for systems that depend on time is highly challenging due to the entanglement of spatiotemporal factors. Weather systems, for instance, contain many interconnected subsystems (1; 2), whereas in multivariate time series, interactions between variables only begin to affect outcomes after a substantial period of time (3; 4). These processes, which naturally occur across a range of spatial and temporal scales, give rise to intricate dynamics that are invisible at the observational level. To address such challenges, existing approaches rely on extracting details using large, multi-layered models, such as TimeMixer++ (5), which typically contains $\sim$ 20 layers in its main body. Due to this parametric density, transformer-based approaches remain data-hungry, while still partially neglecting the dynamics underlying ground-truth observations. Graph Neural Networks (GNNs) offer a more organic way to model foundational processes through sparsely connecting real-world interactions, for example, between particles in a reaction (6).

Though GNNs that use only graph edges excel at representing observational-level processes, they do not consider the dynamics underlying them. To this end, Higher-order GNNs (7; 8) expand the vision of GNNs through embedding knowledge of other graph structures, such as simplexes (9), into reasoning. In forecasting, Hierarchical GNNs (HGNNs) (10; 11; 12) encourage local signals to follow global relationships identified by grouping variables into sub-processes, *e.g.*, a reaction between temperature and humidity. They typically approximate these graphs using a single-pass filter. Yet, due to feature entanglement, these filters associate variables with sub-processes they are not actually part of. One reason for this confounding variable effect is that features may appear close in the shallow embedding space, even though they are unrelated, leading to inaccurate grouping. Such mismatching introduces noise into the sub-process graph, thereby nullifying any potential information gain from further filtering. Methods relying on single-pass filtering (*e.g.* HGNNs), as a consequence, can not fully represent the intricacy of underlying sub-processes.

A critical insight is that deep self-attention on edges can distinguish between two variables even when their embeddings are close together. Though still not separable with a single-pass filter, general form attraction-repulsion equations (13; 14) may use such self-attention to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

disentangle these trends, sorting variables into sub-graphs groups that naturally represent sub-processes and underlying physics. Recent work (15; 16; 17) has focused on using this sorting behavior to address the well-studied oversmoothing problem (18). Combining attraction-repulsion dynamics with self-attention, however, offers a way to find sub-process graphs in an explicit form, without even introducing any additional constraints.

We propose EvolveODE, a Higher-order GNN that discovers these sub-process graphs using a simple assumption that if the embeddings of two variables correlate (attractive case), then they belong to the same sub-process; otherwise (repulsive case), they do not. Their interactions may then be the result of interactions between two sub-processes, rather than between variables. This inductive approach avoids placing unrelated variables into the same group, allowing the creation of graphs in which subsequent filtering continues to yield insights into observational-level dynamics. EvolveODE uncovers and couples various attraction-repulsion equations, creating a system that naturally emulates physics and generates observational phenomena from a bedrock level. EvolveODE, as a result, surpasses current GNN and transformer-based methods in modeling fidelity.

Our contributions made in this work can be summarized as: (1) We propose EvolveODE, a parametrically lightweight Higher-order GNN that organically learns underlying physical processes. (2) We show that stacking level-wise layers in EvolveODE breaks monotonically reduces error. (3) We compare extensively with state-of-the-art methods, evaluating EvolveODE on forecasting over traffic, showing substantial improvements over state-of-the-art methods. EvolveODE also demonstrates robustness to data-scarce scenarios beyond transformers. (4) We evaluate capture of data topology by comparing performance on node classification benchmarks, showcasing consistent gain on both homophilic and heterophilic graphs. (5) We prove convergence properties for our equations.

2 Related work

Learning with Structure. Recent work emphasizes the importance of overlaying geometric bias on top of inductive reasoning. A number of these improve training efficiency by enhancing contextual cues. nGPT (19) trains GPT-3 by projecting data onto a hypersphere, showing a 4-20× faster convergence rate. Hypformer (20) embeds reasoning into a hyperbolic space that captures hierarchical ordering between tokens, halving effective training time. Such contextual cues are also important to the evolution of weather systems, which contain many interconnected subsystems (2; 21). GNNs have proven to be effective at capturing these physical properties. GraphCast (22) represents long-range interactions by adding teleconnections into a multiscale mesh-graph, while SphericalGNN (23) embeds geometric context through spherical convolutions. In spatio-temporal systems, interactions occur between distinct, and often distant, time stamps, leading to the formation of nonlinear dynamics (24; 25). GNNs address this challenge by capturing spatio-temporal interactions either via hypergraphs (26; 27) or physical networks, as in traffic forecasting.

Dynamics on a Continuous Time Scale. In nature, physical processes occur on a continuous time scale, with intricate dynamics shaping outcomes only observable at discrete timestamps. Recent approaches seek to emulate these dynamics by combining continuous-time equations with data-driven learning. Score-based diffusion models (28; 29), for example, simulate image pixels as particles in a continuous-time reaction, where the probability of interaction determines the evolution of perceptual properties. These approaches further allow for flexible sampling regimes in few- or one-shot settings (30). GNNs organically represent physical processes through sparsely coupling components together (31). NeuralODEs (32; 33; 34) solve a system of equations that couple together according to the graph adjacency matrix, as a consequence, overlaying a physical process on top of inductive reasoning. Attraction-repulsion variants (17) disentangle concepts by adding repulsion terms that push apart some features, while also pulling others together through an attraction kernel. However, no work has yet connected attraction-repulsion dynamics with the emulation of underlying system processes. EvolveODE accomplishes this feat, enabling the first marriage of ideas between NeuralODEs and multiscale learning (35).

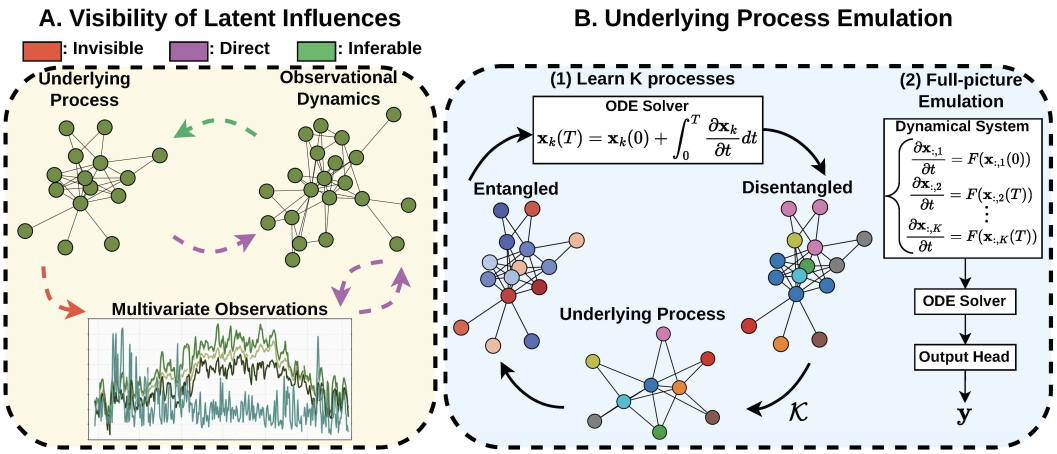


Figure 1: **A.** Overview, using Max Planck weather station data, of the core conceptual challenge EvolveODE addresses. Multivariate signals, *e.g.* temperature and wind speed, arise from and subsequently influence dynamics at the same resolution. These dynamics are a small part of a larger picture in which invisible physical processes affect observations. **B.** (1) Solving an attraction-repulsion equation disentangles features, after which a coarsening kernel $\mathcal{K}$ finds a sub-process graph. (2) Coupling these K equations forms a dynamical system capable of representing the interplay between all sub-processes occurring at the observational level. A task-specific output head subsequently gives the final prediction $\mathbf{y}$.

Multiscale Approaches. Behind all naturally occurring observations are underlying coarse-scale processes governing the evolution of these local phenomena. Numerous works explore the benefits of learning these multiscale relationships in time dependent settings, where dynamics are often time-delayed, *e.g.* transformer-based approaches (36; 37; 38; 5), some of which learn through restrictions of the Fourier decomposition (39; 40; 41). However, progressing such multiscale ideas in spatio-temporal settings remains underexplored, despite HGNNs showing early successes in application areas (12), where underlying sub-process graphs are predetermined and constant. Outside these domains, current HGNNs cannot disentangle local interactions due to their single-pass filtering architecture (11; 42; 43). A performance saturation boundary emerges, where HGNNs may often represent only a single sub-process graph, severely limiting their representational capacity. EvolveODE breaks this saturation boundary by solving an attraction-repulsion equation to disentangle features. The closest existing work to ours is that of HAMP (15) and HDS (16), models which exploit the attraction-repulsion behavior of Allen-Cahn equations to stack multiple layers in their respective settings. EvolveODE, in contrast, uses attraction-repulsion dynamics to learn underlying sub-process graphs explicitly.

3 Methodology

In this section, we develop the theoretical foundations for EvolveODE, a multiscale GNN method that learns and then couples the physical processes underlying observational-level dynamics. When forecasting, for instance, EvolveODE generalizes dynamics learned during the observational window to future samples in the prediction buffer. Figure 1 illustrates the issue with learning only from observational dynamics, along with how EvolveODE seeks to address it. Specifically, by solving an attraction-repulsion equation, EvolveODE sorts variables into sub-process groups, and later coupling of sub-process graphs expands the view to encompass full-picture dynamics. First, we discuss this sorting in Section 3.1, along with other technical points. In Section 3.2, we discuss how sub-process coupling works, and in Section 3.3, we detail the details of full-picture emulation.

3.1 Learning sub-processes through trend disentanglement

Graph topology describes intricate real-world relationships, yet confounding factors entangle their features. This intertwining of concepts means that sub-process graphs are not discernible within a single filtering pass, thereby reducing the fidelity of the learned underlying processes. We make these notions of sub-processes and noise concrete within the following definition:

Definition 3.1 (Sub-processes and Noise)**.** *Given a graph $G = (V, E)$, a sub-process is the interaction between nodes $u \in V$ that form a fully-connected sub-graph $G' = (V', E')$. The graph G' becomes noisy when either $v \in V$ such that $v \notin V'$ is added to G', or $u \in V'$ is removed from G'.*

EvolveODE addresses this challange by disentangling trends with an attraction-repulsion ordinary differential equation, reducing variability arising from sub-process mismatch. These equations exhibit strong self-organization dynamics (14), pulling some features together while pushing others apart, naturally forming clusters that are sub-processes (sub-graphs) from Definition 3.1. Our approach evolves the dynamics of the equation studied in Falcon et al. (13), which describes the interaction between particles in a reaction:

Definition 3.2 (Unconstrained Particle Reaction Equation)**.** *Suppose that there are N features $\mathbf{x}_i(t) \in \mathbb{R}^D$ which encode D spatial attributes at time t of particle i during a reaction. Then the change in these physical properties may be described with kernel W^N and the interaction between i and its neighbors $\mathcal{N}(i)$:*

$$\frac{d\mathbf{x}_i}{dt} = -\sum_{j \in \mathcal{N}(i)} \nabla_{\mathbf{x}} W^N(\mathbf{x}_i(t) - \mathbf{x}_j(t)), \qquad i = 1, \ldots, N, \tag{1}$$

$$\text{where} \quad W^N(\mathbf{y}) = \frac{1}{N} A(\mathbf{y}) - \varepsilon \cdot B\Big(N^{D^{-1}} \mathbf{y}\Big), \qquad \varepsilon \in (0, 1). \tag{2}$$

Specifically, W^N describes particle interaction using self-attention kernels; short-range attraction through A, and long-range repulsion with B, while ε determines the relative strength between these two. Such a range of interactions further encourages cluster formation during convergence to a steady state, allowing EvolveODE to find an ideal disentanglement, *simply by solving Equation 1 forward in time.* Such properties also make EvolveODE robust to both heterophilic and homophilic scenarios. In practice, A and B are matrices found by standard self-attention, and $\nabla_{\mathbf{x}} W^N$ by discretizing with respect to the graph Laplacian.

One issue with the above equation is uncertainty in the number of sub-processes found. Existing attraction-repulsion NeuralODEs (17) include a potential energy term to stabilize these outcomes to one of two possibilities. EvolveODE goes further by adding to Equation 1 an m-well potential energy that encourages the discovery of m sub-processes. This soft constraint increases the number of local minima in the energy landscape, thereby enabling more descriptive dynamics, for instance, by allowing the size of the found sub-processes to vary. This shift in dynamics is therefore necessary to obtain informative sub-process graphs at specific scales - otherwise, features of such graphs are one of two values due to bi-cluster flocking, oversmoothing the sub-process graph. We give the new set of equations below:

Definition 3.3 (Constrained Particle Reaction Equation)**.** *Assume the same conditions as in Definition 3.2. Then, by subtracting the gradient of potential energy function $\mathbf{G}(\mathbf{y})$:*

$$\frac{d\mathbf{x}_i}{dt} = -\sum_{j \in \mathcal{N}(i)} \nabla_{\mathbf{x}} W^N(\mathbf{x}_i(t) - \mathbf{x}_j(t)) - \nabla_{\mathbf{x}} \mathbf{G}(\mathbf{x}_i(t)), \qquad i = 1, \ldots, N, \tag{3}$$

$$\mathbf{G}(\mathbf{y}) = \prod_{q=1}^{m} \Big\| \mathbf{y} - \alpha_q \Big\|_2. \tag{4}$$

Here, $\alpha_q \in \mathbb{R}^D$ are the cluster positions in feature-space. We describe our procedure for finding these in Section 3.1.2. We also scale $\mathbf{G}(\mathbf{y})$ to be bounded above by 1.0, which discourages particles with unbounded energy.

The convergence of equations with the form from Equation 3 for is well established (44). Our theoretical results, as a consequence, place a focus on showing that under the choice of potential energy, Equation 3 satisfies properties such as boundedness and stability. Further, the potential energy itself acts as a constraint. The term ensures steady-states of particle motion can not exist away from α_q; there will always be a force acting on particles otherwise. We only require that their trajectories remain bounded.

After convergence, EvolveODE finds the sub-process graph by applying the kernel $\mathcal{K}$ to group nodes into clusters (variables to sub-processes). We use Graclus clustering to this end (45). The disentanglement of features with an attractive-repulsion equation allows the graph state to become ideal for $\mathcal{K}$ to find the most informative node-to-cluster assignment. An alternative interpretation of disentanglement, in addition to reducing noise in variables to sub-process-specific matching, is to find an ideal Dirichlet energy (18), one that avoids both oversharpening and oversmoothing.

We summarize in Algorithm 1 how EvolveODE uses these attraction-repulsion dynamics to inductively learn K underlying sub-process graphs $G_1, \ldots, G_K$.

Algorithm 1 Learning K Sub-processes

Require: Number of sub-processes K, features $\mathbf{x}_i(t)$ of graph $G_1 = (V_1, E_1)$, step size Δt

$\mathbf{X}(t) \leftarrow [\mathbf{x}(t), \mathbf{0}, \ldots, \mathbf{0}]$
for $k = 1, \ldots, K$ **do**
  $t \leftarrow 0$
  $\mathbf{y}(t) \leftarrow \mathbf{X}_k(t)$
  while not converged **do**
    $\boldsymbol{\Delta}_{i,j} \leftarrow (\mathbf{y}_i(t) - \mathbf{y}_j(t))$ for $(i,j) \in E_k$
    $\frac{d\mathbf{y}_i}{dt} \leftarrow -\sum_{j \in \mathcal{N}(i)} \nabla W^N(\boldsymbol{\Delta}_{i,j}) - \nabla_{\mathbf{x}} \mathbf{G}(\mathbf{y}_i(t))$ for $i \in V_k$
    $\mathbf{y}(t) \leftarrow \mathbf{y}(t) + \frac{d\mathbf{y}}{dt} \cdot \Delta t$
    $t \leftarrow t + \Delta t$
  end while
  $\mathbf{X}_{k+1}(t), G_{k+1} \leftarrow \mathcal{K}(\mathbf{y}(t), G_k)$
end for
return $(\mathbf{X}_1(t), G_1), \ldots, (\mathbf{X}_K(t), G_K)$

3.1.1 m-cluster flocking convergence property

We explore the convergence properties of attraction-repulsion equations with an m-well potential energy term. Particularly, Theorem 1 establishes the asymptotic behavior of Equation 3 as convergence takes place after T steps, showing a Lyapunov Stability condition.

Theorem 3.4 (K-Clusters in the Limit (Lyapunov Stability))**.** *Consider the ODE in Equation 3. Let $\tau > t$, and suppose that $\mathbf{x}_i(t)$ is closest to α_q, such that $\mathbf{x}_i(t) \in B_\delta(\alpha_q)$ for some $\delta > 0$, then there exists $\xi(\delta)$ such that $\mathbf{x}_i(\tau) \in B_{\xi(\delta)}(\alpha_q)$, where one choice is $\xi(\delta) = (1 + \delta m)\delta$ as step size Δt of the numerical solver maps to 0.*

We prove this Theorem in Appendix **A**. Intuitively, the result shows a "is close, stay close" condition, where once the system has begun to converge, node features $\mathbf{x}_i(t)$ will be trapped within the vicinity of an α_q, effectively assigning every such node to the cluster (sub-process) q. Lemma 3.5 shows that for some α_q, the edge-case $\mathbf{x}_i(T) \notin B_\delta(\alpha_q)$ does not hold as T grows large and that particles have bounded trajectories. Our result indicates that all nodes eventually enter a neighborhood around α_q.

Lemma 3.5. *Assuming that q minimizes $\|\mathbf{x}_i(t) - \alpha_q\|_2 > \delta$, there exists a $\varepsilon \in [0, 1]$ that prevents fixed-points in Equation 3 (oversmoothing), and divergent particles (oversharpening), leading to $\|\mathbf{x}_i(t) - \alpha_p\|_2 < \xi(\delta)$, for a $1 \leq p \leq m$, as $t \mapsto \infty$.*

3.1.2 All α_q form a $m-1$ sided polygon on a D-dimensional hypersphere

The proximity between the two clusters raises the question of what ideal convergence might look like. If for some clusters p and q, α_p and α_q are too close, then their basins of attraction could interfere with the trajectory $\mathbf{x}_i(t)$, leading to, for instance, q could have a null-node

assignment. To remedy these complications, we project m evenly spaced points on a D-dimensional hypersphere, which forms a polygon if $m \geq D+1$, for example, an equilateral triangle in two dimensions.

3.2 Full-picture dynamics via coupling all G_1 through to G_K

Running Algorithm 1 finds K underlying sub-process graphs coupling successive graphs G_k and G_{k+1}. In this section, we show that EvolveODE couples all K scales simultaneously, rather than using the inductive coupling of HGNNs, thereby enabling more intricate dynamics in which multiscale views interact.

We write $Q_{i\mapsto j}$ to mean the matrix that couples nodes and clusters between G_i and G_j, where $Q_{k\mapsto k+1}$ are obtained via coarsening with $\mathcal{K}$. Cross-scale coupling is defined as follows:

Definition 3.6 (Higher-Order Connections)**.** *If an integer path $\mathcal{P} = ((i,u),\ldots,(v,j))$ between i and j has values within 1 and K that are either strictly increasing or strictly decreasing, the coupling matrix between G_i and G_j is:*

$$Q_{i\mapsto j} = D \odot \prod_{u,v\in\mathcal{P}} Q_{u\mapsto v}, \tag{5}$$

where D is the normalization matrix mapping nonzero entries to 1.

These matrices are concatenated into an edge index variable, thereby seamlessly integrating with existing GNN forward-call structures. In practice, computing the relation in Equation 5 is done at preprocessing time before the start of every epoch. We also halve the necessary computational time by taking the transpose of $Q_{u\mapsto v}$ to find $Q_{v\mapsto u}$, a relation we show in the proof of Proposition 3.

Proposition 3.7 (Transpose of Coupling Matrices)**.** *For any $1 \leq u, v \leq K$, and coupling matrix $Q_{u\mapsto v}$, then we have:*

$$Q_{u\mapsto v} = Q_{v\mapsto u}^T. \tag{6}$$

3.3 Emulating physical processes with a dynamical system

EvolveODE couples $K-1$ sub-process graphs together with a graph describing observational-level dynamics. Coupling these graphs means EvolveODE also couples its attraction-repulsion equations, yielding a system of coupled equations that captures dynamics beyond those of a stand-alone equation (46). Such a system captures the full picture of the underlying dynamics, which are otherwise invisible at the observational or any single-scale level. We write the feature matrix of this system as $\mathbf{X}(t) = [\mathbf{x}_1(t),\ldots,\mathbf{x}_K(t)] \in \mathbb{R}^{(N_1+\cdots+N_K)\times D}$, where $N_1,\ldots,N_K$ is the sequence for the number of nodes in $G_1,\ldots,G_K$. EvolveODE recomputes kernels A and B, now including full-picture coupling, keeping post-convergence features $\mathbf{X}_2(T),\ldots,\mathbf{X}_K(T)$ while starting $\mathbf{X}_1(0)$ from $t=0$. To write this concretely, we define $\mathbf{Y}(t;T) = [\mathbf{y}_1(t;T),\ldots,\mathbf{y}_K(t;T)]$ such that $\mathbf{Y}_1(0;T) = \mathbf{X}_1(0)$ and $\mathbf{Y}_k(0;T) = \mathbf{X}_k(T)$ for $2 < k \leq K$. From an information-theoretic perspective, such feature revitalization injects previously invisible contextual cues into the disentanglement process.

Let $F(\mathbf{Y}_1(t;T),\ldots,\mathbf{Y}_K(t;T);\alpha_1,\ldots,\alpha_m)$ be the constrained attraction-repulsion ODE from Equation 3, then the fully coupled system at time t is:

$$\frac{d\mathbf{Y}_k}{dt} = F(\mathbf{Y}_1(t;T),\mathbf{Y}_2(t;T),\ldots,\mathbf{Y}_K(t;T);\alpha_{N_{k-1}},\ldots,\alpha_{N_k}), \qquad 1 \leq k \leq K. \tag{7}$$

We solve this system again, using T solver steps, and pass $\mathbf{Y}_1(T;T)$ through an output head specific to the downstream task. For instance, in forecasting settings, a mapping onto the output horizon (43).

Table 1: MAE across forecast horizons, averaged over input buffer sizes 12, 24, and 48.

Model	PEMS03			PEMS04			PEMS08		
	3	6	9	3	6	9	3	6	9
TimeMixer++ (5)	0.143	0.153	0.163	0.160	0.172	0.180	0.147	0.157	0.166
TimerXL (47)	0.144	0.158	0.171	0.165	0.179	0.194	0.151	0.165	0.178
iTransformer (48)	0.142	0.156	0.169	0.163	0.177	0.192	0.148	0.161	0.173
MAGE (49)	0.143	0.154	0.163	0.161	0.171	0.177	0.147	0.156	0.163
CITRUS (50)	0.149	0.163	0.175	0.166	0.173	0.188	0.152	0.164	0.173
EvolveODE	**0.139**	**0.151**	**0.160**	**0.154**	**0.164**	**0.172**	**0.141**	**0.153**	**0.160**
Gain (%)	**+2.1%**	**+1.3%**	**+1.8%**	**+3.8%**	**+4.1%**	**+2.8%**	**+4.1%**	**+1.9%**	**+1.8%**

4 Experiments

We evaluate EvolveODE's effectiveness in addressing task-specific challenges. In short-term forecasting, EvolveODE extrapolates spatio-temporal relationships learned from a small data buffer into future scenarios. At the same time, in node classification, EvolveODE infers missing class labels from an input graph by extracting information from its topology. Consistently across these tasks, we seek to determine the extent to which EvolveODE can approximate ground-truth dynamics and under what settings emulation is no longer possible.

4.1 Experimental setup

We describe the baselines, datasets, and experimental procedures in this section.

Datasets. To evaluate the effectiveness of inferring missing labels in node classification, we compare on both homophilic graphs, namely Cora and Citeseer, alongside heterophilic graphs; Texas, Wisconsin, Squirrel. In the former set, neighbors predominantly share the same class label, whereas in the latter, they tend to differ. Such experiments evaluate EvolveODE's adaptability to potentially adverse settings. We also evaluate against traffic forecasting benchmarks PEMS03, PEMS04, and PEMS08. All GNN methods have access to their road network graphs. We use a 0.8, 0.1, and 0.1 training, validation, and test split.

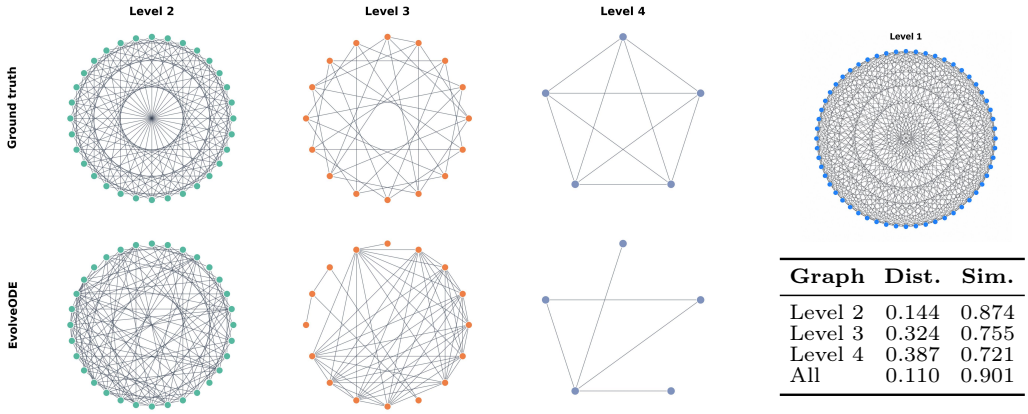
Baselines. We compare against state-of-the-art transformer baselines, namely TimerXL (47), TimeMixer++ (5), TinyTimeMixer (51) and iTransformer (48). We add to these comparisons multiscale GNN forecasting methods MAGE (49) and CITRUS (50). For the node classification task, we compare against continuous-time transformer methods, including GrokFormer (52), DIFFormer (53), and AdvDIFFormer (54). We also include NeuralODEs with attraction-repulsion dynamics, namely ACMP (17), HAMP-I (15), and HDS (16), the latter two of which explicitly incorporate first and second-order dynamics.

Reproducibility. We cover the exact experimental parameters. To run all forecasting methods, we use their optimal hyperparameter settings, if available, with learning and dropout rates of 0.0005 and 0.5, respectively. For node classification, we fine-tune each baseline with respect to learning rate and weight decay using a grid search. These parameters are reported via code link. In traffic forecasting, EvolveODE uses a system of 5 equations, where equation k has $\lfloor \frac{N}{2^k} \rfloor$ variables, whereas in weather forecasting. To solve the equations, EvolveODE uses 5 forward Euler steps with $T = 5$. We used a NVIDIA H200 GPU.

4.2 Evaluation

We compare EvolveODE against various state-of-the-art baselines on traffic forecasting and node classification tasks. Appendix **C** reports the outcomes of experiments on autoregressive weather forecasting and ablations that evaluate the efficacy of solving the m-well ODE.

Short-horizon traffic forecasting. We give our results from the short-horizon traffic forecasting experiments in Table 1. EvolveODE consistently obtains the lowest MAE by comfortable margins, ranging from around 1.5% to 4%. This outcome holds for both the parametrically dense transformer methods and multiscale GNNs, providing some initial demonstration as to the benefits of adding contextual cues to disentangle features. Specifically,



Graph	Dist.	Sim.
Level 2	0.144	0.874
Level 3	0.324	0.755
Level 4	0.387	0.721
All	0.110	0.901

Figure 2: Synthetic experiment evaluating how well EvolveODE discovers underlying processes. Shown are the ground truth underlying sub-process graphs for a system of coupled oscillators (see Appendix **B**), along with EvolveODE's indirect approximations of them. The fidelity of these are evaluated using topological distance and spectral similarity metrics.

Table 2: MAE and RMSE across forecast horizons with a buffer size of 96.

Model	PEMS03				PEMS08			
	72		96		72		96	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
TimeMixer++ (5)	0.256	**0.372**	0.276	0.418	0.259	0.385	0.258	0.386
iTransformer (48)	0.289	0.444	0.383	0.557	0.296	0.442	0.318	0.483
TimerXL (47)	0.497	0.698	0.364	0.530	0.443	0.628	0.554	0.770
CITRUS (50)	0.284	0.408	0.298	0.426	0.856	1.015	0.266	0.373
EvolveODE	**0.249**	0.375	**0.262**	**0.395**	**0.233**	**0.345**	**0.241**	**0.354**
Gain (%)	**+2.8%**	**-0.7%**	**+5.1%**	**+5.6%**	**+10.0%**	**+9.0%**	**+6.9%**	**+8.1%**

our results show that EvolveODE effectively does such disentanglement under settings when the input buffer contains a shallow description of previous temporal trends.

Long-horizon traffic forecasting. In Table 2, we report results over long traffic forecasting horizons when using a large input buffer of size 96. EvolveODE achieves large performance gains across almost all categories, showing that the learned dynamics generalises across a wide horizon. This consistency further demonstrates that EvolveODE can disentangle intricate temporal patterns, even when accounting for time-delayed dynamics that strongly influence long-horizon settings. Our results show the efficacy of learning underlying processes.

Synthetic experiment on discovering known latent sub-processes. In Figure 2, we evaluate the ability of EvolveODE to discover known sub-processes by training on the solution of a system of coupled Kuramoto oscillators (46) that comprises of four graphs with distinct granularity. We supply the graph of the highest resolution to EvolveODE, and evaluate the extent to which the remaining graphs are uncovered. Appendix **C** provides further details. We observe a high-degree of reconstruction for the Level 2 sub-process graph, with similarity and distance metrics around 0.87 and 0.144, respectively. These values tend to depreciate, albeit slowly, with level index. However, the reconstruction of the fourth level, being inherently the most challenging, still displays high fidelity both visually and through strong quantitative metrics. These results demonstrate that EvolveODE uncovers sub-processes to a high degree of accuracy, even in systems with intricate coupling topology.

Node classification. The outcomes for our node classification experiments are shown in Table 3. Due to information bottleneck in graphs from these datasets, we only use an EvolveODE with no more than two equations, and interpret outcomes as an indication of how well EvolveODE can disentangle nodes directly into class bins. While EvolveODE is

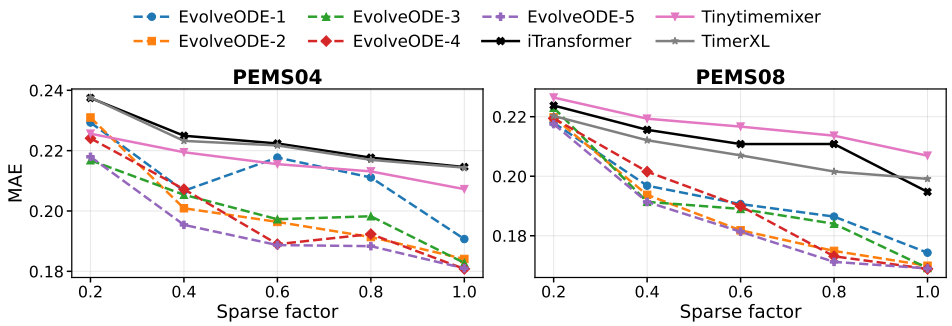


Figure 3: Test set MAE after training on a subset of data, consisting of $0.2, 0.4, 0.6, 0.8$ and 1.0 percentage of the original training data. EvolveODE-n has n extra equations.

Table 3: Node classification accuracy (%) on homophily and heterophily datasets.

Category	Model	Homophily		Heterophily		
		Cora	Citeseer	Texas	Wisconsin	Squirrel
Transformer	DIFFformer (53)	80.8 ± 0.9	70.9 ± 1.3	80.5 ± 3.1	77.3 ± 2.0	39.7 ± 2.9
	AdvDIFFormer (54)	79.4 ± 1.1	66.1 ± 1.0	77.8 ± 3.2	77.9 ± 2.9	38.1 ± 2.3
	GrokFormer (52)	73.1 ± 0.5	63.6 ± 0.3	80.8 ± 2.6	77.2 ± 1.4	37.1 ± 1.4
NeuralODE	HAMP-I (15)	80.5 ± 0.6	67.6 ± 0.3	77.8 ± 1.1	71.4 ± 2.9	37.7 ± 1.5
	HDS (16)	81.2 ± 0.9	70.4 ± 0.7	81.1 ± 1.2	75.8 ± 1.9	38.9 ± 0.9
	ACMP (17)	82.1 ± 1.2	71.3 ± 0.6	81.1 ± 2.9	78.4 ± 0.3	34.7 ± 0.4
Ours	EvolveODE-1	82.9 ± 0.5	71.3 ± 0.4	79.6 ± 3.8	81.9 ± 0.9	45.7 ± 1.8
	EvolveODE-2	**84.2 ± 0.2**	**71.6 ± 0.6**	**82.7 ± 2.6**	**83.8 ± 1.5**	**46.5 ± 0.9**
Gain (%)		**+2.6%**	**+0.4%**	**+1.9%**	**+6.9%**	**+7.2%**

beneficial on homophilic graphs, increasing accuracy by 0.4% and 2.6%, the difference is most stark on heterophilic graphs, where accuracy increases by 1.9% to 7.2%. EvolveODE also improves over the HDS and HAMP methods, which also incorporate both first and second order dynamics. Learning a latent sub-process instead of using only existing structure, appears to have verifiable benefits. Such higher-order views evidently reveal critical topology.

Evaluating under data-scarce settings. We study the data-hungriness of EvolveODE, by investigating to what extent a system comprising of more equations requires more data to be effective. Our results also include transformers to contrast their stability under data sparsity with that of EvolveODE. We present our results in Figure 3. The system of 5 equations immediately achieves the lowest MAE, even under a sparse factor of 0.2. This result emerges due to a 0.5 reduction factor in graph size, which jointly reduces necessary representational capacity to learn new relationships. Moreover, as also shown in Appendix **C**, adding equations to the system monotonically reduces error, solidifying our claims around the breaking of saturation boundaries. Further, EvolveODE continues to outperform layer-by-layer approaches across all settings, including models that excel under data-scarce regimes, such as TinyTimeMixer. EvolveODE has an advantage in terms of parametrically lightweight design, using only a single round of self-attention. Due to the separation of features, EvolveODE discovers intricate relationships without relying on deep extraction.

5 Conclusion

In this work, we propose EvolveODE, a Higher-order GNN that, unlike approaches ensuring performance through the stacking of many layers, directly emulate the latent processes underlying observational dynamics. By sorting using attraction-repulsion equations, EvolveODE approximates, from a bedrock level, the governing equations for a set of observations with origins describable through physical laws. Across various tasks, EvolveODE achieves state-of-the-art performance, opening the door to new ideas in computational modeling of real-world domains that are, for instance, less data hungry than many existing layer-by-layer methods.

6 AI Use Statement

In this work, Generative AI has been used to format tables and create figures displaying empirical results. Further, the authors use this tool for setting up experiments and implementing baselines. AI has not been used in other areas of the research workflow.

7 Ethics statement

The use cases of the work do not immediately pose an ethical risk

8 Reproducibility statement

The paper provides a code link in the abstract, which contains .sh files used to run all experiments, along with a README file containing instructions for readers on how to run their own experiments. In addition, the paper discloses key hyperparameters in the reproducibility section. The main paper also provides in detail step-by-step methodological details, such as in Algorithm 1.

References

[1] J. David Neelin, Mojib Latif, and F.-F. Jin. Dynamics of coupled ocean-atmosphere models: The tropical problem. *Annual Review of Fluid Mechanics*, 26(1):617–659, 1994.

[2] Jacob Bjerknes. Atmospheric teleconnections from the equatorial pacific. *Monthly Weather Review*, 97(3):163–172, 1969.

[3] Swinda K. J. Falkena, Courtney Quinn, Jan Sieber, and Henk A. Dijkstra. A delay equation model for the atlantic multidecadal oscillation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2246):20200659, 2021.

[4] I. N. Kiselev, I. R. Akberdin, and F. A. Kolpakov. Delay-differential seir modeling for improved modelling of infection dynamics. *Scientific Reports*, 13(1):13439, 2023.

[5] Shiyu Wang, Jiawei Li, Xiaoming Shi, Zhou Ye, Baichuan Mo, Wenze Lin, Shengtong Ju, Zhixuan Chu, and Ming Jin. Timemixer++: A general time series pattern machine for universal predictive analysis. In *International Conference on Learning Representations*, 2025.

[6] Hao Wu, Changhu Wang, Fan Xu, Jinbao Xue, Chong Chen, Xian-Sheng Hua, and Xiao Luo. Pure: Prompt evolution with graph ode for out-of-distribution fluid dynamics modeling. *Advances in Neural Information Processing Systems*, 37:104965–104994, 2024.

[7] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[8] Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. Subgraph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 18583–18595, 2020.

[9] Yiming Huang, Yujie Zeng, Qiang Wu, and Linyuan Lü. Higher-order graph convolutional network with flower-petals laplacians on simplicial complexes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12653–12661, 2024.

[10] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2083–2092, 2019.

[11] Andrea Cini, Danilo Mandic, and Cesare Alippi. Graph-based time series clustering for end-to-end hierarchical forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 8985–8999. PMLR, 2024.

[12] Joel Oskarsson, Tomas Landelius, Marc Peter Deisenroth, and Fredrik Lindsten. Probabilistic weather forecasting with hierarchical graph neural networks. In *Advances in Neural Information Processing Systems*, 2024.

[13] Carles Falcó, Ruth E Baker, and José A Carrillo. A nonlocal-to-local approach to aggregation-diffusion equations. *SIAM Review*, 67(2):353–372, 2025.

[14] Martin Burger, Marco Di Francesco, Simone Fagioli, and Angela Stevens. Sorting phenomena in a mathematical model for two mutually attracting/repelling species. *SIAM Journal on Mathematical Analysis*, 50(3):3210–3250, 2018.

[15] Yixuan Ma, Kai Yi, Pietro Liò, Shi Jin, and Yu Guang Wang. How particle system theory enhances hypergraph message passing. In *Advances in Neural Information Processing Systems*, 2025.

[16] Jielong Yan, Yifan Feng, Shihui Ying, and Yue Gao. Hypergraph dynamic system. In *The Twelfth International Conference on Learning Representations*, 2024.

[17] Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. Acmp: Allen-cahn message passing with attractive and repulsive forces for graph neural networks. In *International Conference on Learning Representations*, 2023.

[18] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[19] Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *International Conference on Learning Representations (ICLR)*, 2025.

[20] Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. Hypformer: Exploring efficient transformer fully in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3770–3781, 2024.

[21] Zhengyu Liu and Michael Alexander. Atmospheric bridge, oceanic tunnel, and global climatic teleconnections. *Reviews of Geophysics*, 45:RG2005, 2007.

[22] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677), 2023.

[23] Kenghong Lin, Xutao Li, Yunming Ye, Shanshan Feng, Baoquan Zhang, Guangning Xu, and Ziye Wang. Spherical neural operator network for global weather prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(6):4899–4913, 2024.

[24] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR)*, 2018.

[25] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3634–3640, 2018.

[26] Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. Fouriergnn: Rethinking multivariate time series forecasting from a pure graph perspective. In *Advances in Neural Information Processing Systems*, 2023.

[27] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[28] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2021.

[29] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[30] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 23–29 Jul 2023.

[31] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[32] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

[33] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: Graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1407–1418, 2021.

[34] Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. GRAND++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2022.

[35] Zewen Liu, Xiaoda Wang, Bohan Wang, Zijie Huang, Carl Yang, and Wei Jin. Graph odes and beyond: A comprehensive survey on integrating differential equations with graph neural networks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6118–6128, 2025.

[36] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler-Canseco, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6989–6997, 2023.

[37] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

[38] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y. Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International Conference on Learning Representations*, 2024.

[39] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27268–27286. PMLR, 2022.

[40] Yifan Hu, Peiyuan Liu, Peng Zhu, Dawei Cheng, and Tao Dai. Adaptive multi-scale decomposition framework for time series forecasting. *arXiv preprint arXiv:2406.03751*, 2024.

[41] Yulong Wang, Yushuo Liu, Xiaoyi Duan, and Kai Wang. Filterts: Comprehensive frequency filtering for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[42] Yirong Chen, Ziyue Li, Wanli Ouyang, and Michael D. Lepech. Adaptive hierarchical spatiotemporal network for traffic forecasting. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6. IEEE, 2023.

[43] Yicheng Luo, Bowen Zhang, Zhen Liu, and Qianli Ma. Hi-patch: Hierarchical patch gnn for irregular multivariate time series. In *Forty-second International Conference on Machine Learning*, 2025.

[44] Xiyang Luo and Andrea L. Bertozzi. Convergence of the graph allen–cahn scheme. *Journal of Statistical Physics*, 167:934–958, 2017.

[45] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

[46] Steven H Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1-4):1–20, 2000.

[47] Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. *International Conference on Learning Representations*, 2025.

[48] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

[49] Jiaming Ma, Binwu Wang, Guanjun Wang, Kuo Yang, Zhengyang Zhou, Pengkun Wang, Xu Wang, and Yang Wang. Less but more: Linear adaptive graph learning empowering spatiotemporal forecasting. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[50] Aref Einizade, Fragkiskos D. Malliaros, and Jhony H. Giraldo. Continuous product graph neural networks. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[51] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Advances in Neural Information Processing Systems*, 2024.

[52] Guoguo Ai, Guansong Pang, Hezhe Qiao, Yuan Gao, and Hui Yan. Grokformer: Graph fourier kolmogorov-arnold transformers. *International Conference on Machine Learning*, 2024.

[53] Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

[54] Qitian Wu, Chenxiao Yang, Kaipeng Zeng, and Michael Bronstein. Supercharging graph transformers with advective diffusion. In *International Conference on Machine Learning*, 2025.

[55] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35:9881–9893, 2022.

A Proofs on convergence properties and coupling matrices

We previously proved theoretical statements in this section regarding the convergence of the constrained system in Equation 3 of the main paper. Such properties allow learning K equations, which are later coupled into a dynamical system.

Theorem 1 (K-Clusters in the Limit (Lyapunov Stability))**.** *Consider the ODE in Equation 3. Let $\tau > t$, and suppose that $\mathbf{x}_i(t)$ is closest to α_q, such that $\mathbf{x}_i(t) \in B_\delta(\alpha_q)$ for some $\delta > 0$, then there exists $\xi(\delta)$ such that $\mathbf{x}_i(\tau) \in B_{\xi(\delta)}(\alpha_q)$, where one choice is $\xi(\delta) = (1 + \delta m)\delta$ as step size Δt of the numerical solver maps to 0.*

Proof. If $\mathbf{x}_i(t) \in B_\delta(\alpha_q)$, then $\|\mathbf{x}_i(t) - \alpha_q\|_2 \leq \delta$. We then take a $\tau > t$. Using the fundamental theorem of calculus,

$$\mathbf{x}_i(\tau) \;=\; \mathbf{x}_i(t) \;+\; \int_t^\tau \frac{\partial \mathbf{x}_i}{\partial s}\, ds,$$

we aim to derive a $\xi(\delta)$ such that $\delta \leq \xi(\delta)$, and where q is the minimizer of $\|\mathbf{x}_i(t) - \alpha_p\|_2$:

$$\begin{aligned}
\Big\|\mathbf{x}_i(\tau) - \alpha_q\Big\|_2 &= \Big\|\mathbf{x}_i(t) - \alpha_q + \int_t^\tau \frac{d\mathbf{x}_i(s)}{ds}\, ds\Big\|_2 \\
&\leq \Big\|\mathbf{x}_i(t) - \alpha_q\Big\|_2 + \Big\|\int_t^\tau \frac{d\mathbf{x}_i(s)}{ds}\, ds\Big\|_2 \\
&\leq \delta + \int_t^\tau \Big\|\frac{d\mathbf{x}_i(s)}{ds}\Big\|_2 ds \\
&= \delta + \int_t^\tau \Big\|\sum_{j\in\mathcal{N}(i)} \omega_{ij}\cdot\big(\mathbf{x}_i(s) - \mathbf{x}_j(s)\big) - \nabla_{\mathbf{x}}\mathbf{G}(\mathbf{x}_i(s))\Big\|_2 ds. \\
&\leq \delta + \int_t^\tau \sum_{j\in\mathcal{N}(i)} |\omega_{ij}|\cdot\Big\|\mathbf{x}_i(s) - \mathbf{x}_j(s)\Big\|_2 + \Big\|\nabla_{\mathbf{x}}\mathbf{G}(\mathbf{x}_i(s)\Big\|_2 ds \\
&\leq \delta + C\cdot(\tau - t) + \int_t^\tau \Big\|\nabla_{\mathbf{x}}\mathbf{G}(\mathbf{x}_i(s)\Big\|_2 ds
\end{aligned} \tag{8}$$

Here, C is a bounded constant, since $\omega_{i,j} \in [1, -\varepsilon]$. To bound the potential energy term, we use the fact that

$$\Big\|\nabla_{\mathbf{x}}\mathbf{G}(\mathbf{x}_i(s)\Big\|_2 \leq 2\sum_{q=1}^m \Big\|\mathbf{x}_i(t) - \alpha_q\Big\|_2 \prod_{q=1}^m \Big\|\mathbf{x}_i(t) - \alpha_q\Big\|_2 \tag{9}$$

And to bound the product term:

$$\begin{aligned}
\prod_{1\leq p\leq N_{i+1}} \Big\|\mathbf{x}_i(t) - \alpha_q\Big\|_2 &\leq \Big\|\mathbf{x}_i(t) - \alpha_q\Big\|_2 \cdot \prod_{1\leq p\leq N_{i+1}, p\neq q} \Big\|\mathbf{x}_i(t) - \alpha_p\Big\|_2, \\
&\leq \delta \cdot \prod_{1\leq p\leq N_{i+1}, p\neq q} \Big\|\mathbf{x}_i(t) - \alpha_q + \alpha_q - \alpha_p\Big\|_2, \\
&\leq \delta \cdot \prod_{1\leq p\leq N_{i+1}, p\neq q} \Big\|\mathbf{x}_i(t) - \alpha_q\Big\|_2 + \Big\|\alpha_q - \alpha_p\Big\|_2, \\
&\leq \delta \cdot \prod_{1\leq p\leq N_{i+1}, p\neq q} (\delta + \sum_{1\leq l\leq |p-q|} \Big\|\alpha_l - \alpha_{l+1}\Big\|_2) \\
&\leq \delta \cdot \prod_{1\leq p\leq N_{i+1}, p\neq q} (\delta + \frac{\pi}{m}|p-q|)
\end{aligned} \tag{10}$$

The second-to-last line follows from the triangle inequality on the shortest path between p and q, whereas the last line follows from substituting the distance between α_l and α_{l+1}, which is constant and equal to $\sin(\frac{\pi}{m}) \approx \frac{\pi}{m}$ for large m. Then the sequence converges to $\delta(\delta + \frac{\pi}{m}\cdot 1)\ldots(\delta + \frac{\pi}{m}(m-1)) \approx \delta^{n'+1}\cdot\pi^{n''}$.

For the summation, using the same method as in the product term:

$$\begin{aligned}\sum_{q=1}^{m}\left\|\mathbf{x}_i(t)-\alpha_q\right\|_2 &\leq \delta+\sum_{p\neq q}\sum_{1\leq l\leq|p-q|}\left\|\alpha_l-\alpha_{l+1}\right\|_2\\ &\leq \delta+\frac{\pi}{m}(1+2+\ldots m-1)\\ &= \delta+\frac{\pi}{2}(m-1)\\ &\approx \frac{\pi}{2}(m-1)\end{aligned} \tag{11}$$

The total expression reduces to

$$\left\|\mathbf{x}_i(\tau)-\alpha_q\right\|_2 \leq \delta+C(\tau-t)+\delta^{n'+1}\cdot\pi^{n''+1}\cdot(m-1) \tag{12}$$

For sufficiently small δ and $\Delta t=\tau-t$, we have that the entire expression is bounded by $\delta+\delta^2 m=(1+\delta m)\delta<1$.

We show that under step length $l=\tau-t$, such that, by continuity $\mathbf{x}_i(\tau)\in B_{\xi(\delta)}(\alpha_q)$, that $\xi(\delta)$ is close to δ:

□

We show in the next lemma that the above theorem can be applied to any $\mathbf{x}_i(t)$; that is, the entire system converges to areas around α_q for $1\leq q\leq m$.

Lemma 2. *Assuming that q minimizes $\|\mathbf{x}_i(t)-\alpha_q\|_2>\delta$, there exists a $\varepsilon\in[0,1]$ that prevents fixed-points in Equation 3 (oversmoothing), and divergent particles (oversharpening), leading to $\|\mathbf{x}_i(t)-\alpha_p\|_2<\xi(\delta)$, for a $1\leq p\leq m$, as $t\mapsto\infty$.*

Proof. If $\mathbf{x}(t)$ is a fixed point, then for all i:

$$\sum_{j\in\mathcal{N}(i)}\omega_{i,j}\cdot(\mathbf{x}_i(t)-\mathbf{x}_j(t))=-\prod_{1\leq q\leq N_{i+1}}\left\|\mathbf{x}_i(t)-\alpha_q\right\|_2\leq 0 \tag{13}$$

We use the fact that $\omega_{i,j}=A_{i,j}-\varepsilon B_{i,j}$, for attraction and repulsion kernels A and B, to simplify the leftmost inequality:

$$\begin{aligned}\sum_{j\in\mathcal{N}(i)}\omega_{i,j}\cdot(\mathbf{x}_i(t)-\mathbf{x}_j(t)) &= \sum_{u\in\mathcal{N}(i)}A_{i,u}\cdot(\mathbf{x}_i(t)-\mathbf{x}_u(t))-\varepsilon\sum_{v\in\mathcal{N}(i)}B_{i,v}\cdot(\mathbf{x}_i(t)-\mathbf{x}_v(t)),\\ &= \mathbb{E}_{(i,u)}\Big[\mathbf{x}_i(t)-\mathbf{x}_u(t)\Big]-\varepsilon\cdot\mathbb{E}_{(i,v)}\Big[\mathbf{x}_i(t)-\mathbf{x}_v(t)\Big],\\ &= (\mathbf{x}_i(t)-\widehat{\mathbf{x}})-\varepsilon\cdot(\mathbf{x}_i(t)-\bar{\mathbf{x}})\leq 0.\end{aligned} \tag{14}$$

Therefore, there can only exist fixed points away from α_q if $(\mathbf{x}_i(t)-\widehat{\mathbf{x}})\leq\varepsilon\cdot(\mathbf{x}_i(t)-\tilde{\mathbf{x}})$, meaning that for some ε, the attractive force is stronger than the repulsion one. Similarly, an ε exists that stops particles from having an unbounded potential energy. These observations preclude the existence of stable and unstable fixed points. The only minima in the energy landscape are then α_q, obtaining the result.

□

We prove below that the transpose of a coupling matrix $Q_{v\mapsto u}$, which couples nodes within graphs G_v to nodes within G_u, is the transpose of $Q_{u\mapsto v}$, allowing an effective $\mathcal{O}(1)$ way to determine bidirectional coupling:

Proposition 3 (Transpose of Coupling Matrices)**.** *Suppose that $|V(\mathcal{M})|=K$, and that $Q_{i\mapsto i+1}$, for $1\leq i\leq K-1$ are known incident matrices. For $i,j,l\in V(\mathcal{M})$, then:*

$$Q_{u\mapsto v}=Q_{v\mapsto u}^{T} \tag{15}$$

for any $u,v\in V(\mathcal{M})$.

Proof. Suppose that there exists a shortest path P_{uv} on $\mathcal{M}$ between u and v, and conversely such a path between v and u, P_{vu}. Then, $Q_{i\mapsto i+1} = Q_{i+1\mapsto i}^T$ by orthogonality, and:

$$\begin{aligned} Q_{u\mapsto v}^T &= (\prod_{(s,s')\in P_{uv}} Q_{s\mapsto s'})^T \\ &= \prod_{(s',s)\in P_{vu}} Q_{s\mapsto s'}^T \\ &= \prod_{(s',s)\in P_{vu}} Q_{s'\mapsto s} \\ &= Q_{v\mapsto u}. \end{aligned}$$

Here, we have used $(AB)^T = B^T \cdot A^T$ and the axiom $Q_{i\mapsto l} = Q_{i\mapsto j} \cdot Q_{j\mapsto l}$.. □

B Additional Methodological Details

We provide here additional discussions surrounding key aspects of EvolveODE, namely the exact form of the hyper-spherical embeddings, and further details on convergence.

Hyperspherical Projection. We supplement the discussion in the main paper surrounding the calculation of α_q. These are m points spaced evenly on a D dimension al hypersphere, where m is the number of roots in the force constraint $\nabla\mathbf{G}$, and D is the hidden dimension. The projection is written using $\theta_q = -\pi + \frac{2\pi(q-1)}{m}$ and $\alpha_{q,d}$ as the dth entry of α_q:

$$\phi_{q,d} = \begin{cases} \big((\theta_q + \pi)d\big) \bmod \pi, & d = 1, \dots, D-2, \\ \theta_q, & d = D-1. \end{cases} \tag{16}$$

$$\alpha_{q,d} = \left(\prod_{\ell=1}^{d-1} \sin(\phi_{q,\ell})\right) \cos(\phi_{q,d}), \qquad d = 2, \dots, D-1. \tag{17}$$

By selecting such evenly spaced points, EvolveODE enhances the well-posedness of sorting variables to sub-processes.

Additional discussion on system convergence. We also supplement in this section our discussion on the convergence of Equation 2 and 4. The former equation converges, since (1) the particles decrease with respect to the total energy $\Phi(\mathbf{x}(t))$. This principle states that particle trajectory always takes the lowest cost path:

$$\frac{\partial \mathbf{x}_i}{\partial t} = -\frac{\partial \Phi}{\partial \mathbf{x}_i} \tag{18}$$

Further, we have, by the chain rule:

$$\frac{\partial \Phi}{\partial t} = \frac{\partial \Phi}{\partial \mathbf{x}_i}\frac{\partial \mathbf{x}_i}{\partial t} = -\left\| \frac{\partial \Phi}{\partial \mathbf{x}_i} \right\|_2 \leq 0. \tag{19}$$

If Φ also satisfies $\Phi(\mathbf{x}(t)) \geq 0$, as all energies do, then the system converges, and the solution is bounded. That is, we do not have a divergent particle, for instance, $\mathbf{x}_i(t) \mapsto \infty$ as $t \mapsto \infty$. Equation 3 then trivially converges if the new energy $\widetilde{\Phi} = \Phi + \mathbf{G}$ is also positive semi-definite, which holds since $\mathbf{G}$ in this case is positive semi-definite.

Synthetic Dynamical System. We cover the procedure of generating a time dependent signal $Y(t)$ from a system of equations, each representing the observational and underlying sub-processes. Our system contains 4 sets of such equations, where the graph of highest resolution corresponds to a known system of interactions. EvolveODE is given this graph as input, and must then infer the others indirectly by attempting to train on signal $Y(t)$.

These graphs $G_1, \dots, G_4$ consist of $64, 32, 16$ and 5 nodes. Internally, two nodes i, j coupling variables of equation set $1 \leq k \leq 4$ share an edge if $i \bmod q_k \equiv j \mod q_k$. Here, q_k is an

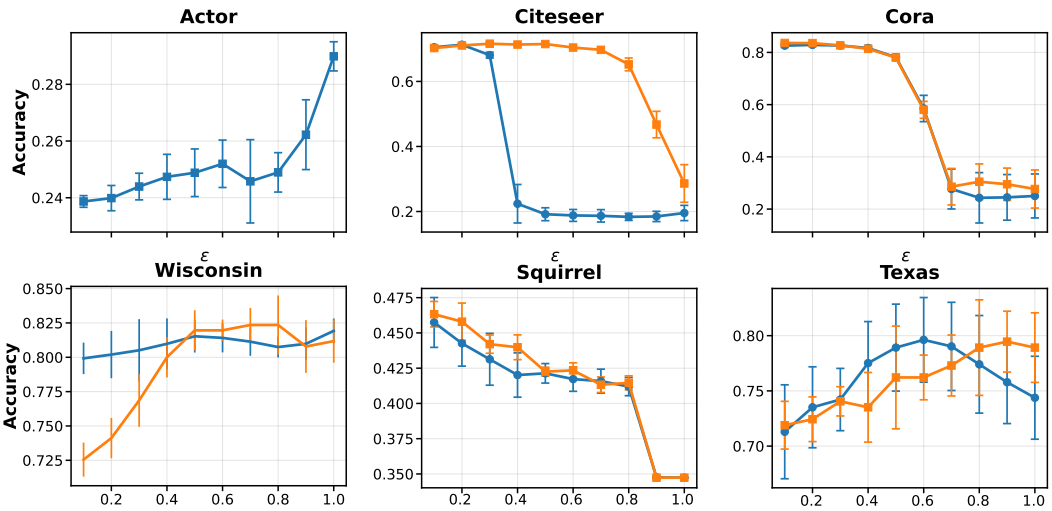


Figure 4: Accuracy on node classification datasets, both homophilic and heterophilic, when the strength of repulsion ε varies.

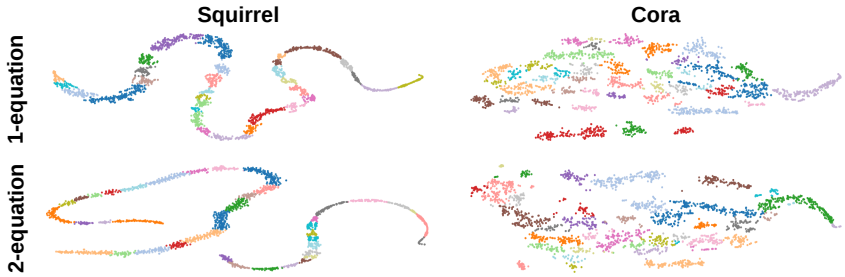


Figure 5: The t-SNE projections of EvolveODE features embeddings prior to output head mapping on Squirrel (heterophilic) and Cora (homophilic) graphs. We show EvolveODE the difference in the learned structure between a system of one and two equations.

integer determining coupling density, and are set to $4, 3, 2$ and 1. External coupling between i in G_k and j in $G_{k'}$, for disparate integers k and k', exist if $(i + N_k) \bmod p \equiv (j + N_{k'}) \bmod p$, where $p = 3$ for all levels.

The signal itself arises from a system of coupled Kuramoto oscillators:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{1}{|\mathcal{N}(i)|} \sum_{j \in |\mathcal{N}(i)|} W_{i,j} \sin(\theta_j - \theta_i) \tag{20}$$

Here, coupling matrix $W \in \mathbb{R}^{117 \times 117}$ $(117 = 64 + 32 + 16 + 5)$ is a matrix consisting of cross coupling strengths, and $\omega_i \in \mathbb{R}$ are inherent frequencies of each node. Both these quantities are randomly initialized with normal distributions.

We solve this system forward in time, taking $Y(t) = \theta(t) + z(t)$, adding the noise vector $z(t) \sim \mathcal{N}(0, 0.3 \cdot I)$ to break periodicity and create a more challenging learning task.

C Experiments

Visualizing structure with t-SNE plots. Figure 5 showcases the converged features of the original graph after t-SNE projection onto two dimensions. EvolveODE shows a clear separation of features in both heterophilic and homophilic datasets, with improvements in structural complexity from an additional equation evident in the Squirrel dataset.

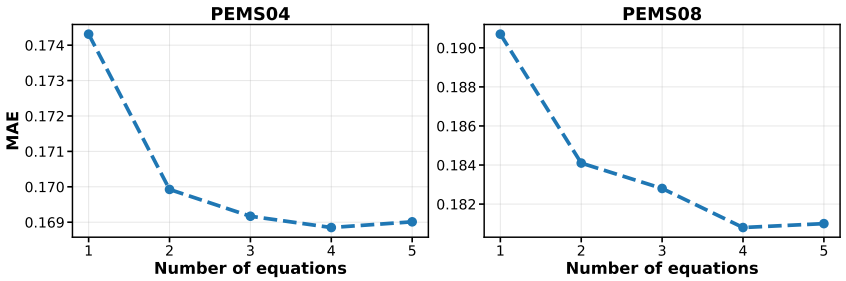


Figure 6: The effect of the number of equations EvolveODE uses on MAE. The input buffer size and prediction buffer length are both 12.

Table 4: Computational and memory statistics.

Model	GFLOPs/step	TFLOPs/epoch	Runtime/epoch	Peak RAM	Peak GPU Mem.
DCRNN	4210.28	930.47	623.01 s	4.38 GB	15.10 GB
STGCN	343.79	75.98	26.26 s	1.88 GB	5.77 GB
EvolveODE-1	59.14	13.07	27.46 s	2.32 GB	7.53 GB
EvolveODE-2	105.32	23.28	56.59 s	2.71 GB	7.57 GB
EvolveODE-5	154.54	34.15	120.61 s	2.86 GB	7.58 GB

Verifying saturation boundary claims. We further investigate the findings of the main paper, shown in Figure 3, showing that EvolveODE does break the saturation boundary shown by HGNNs. In Figure 6, we show the effects of varying the number of equations in the dynamical system of EvolveODE. The error metric decreases monotonically until a system of 5 equations. At this point, the graph size is small enough that applying a coarsening kernel does not yield additional information gain. We call this the finite split problem. We discuss this further in the limitations and future work sections.

Changing repulsion terms. Figure 4 shows the effect of changing the repulsion strength of the particle physics kernel. In doing so, nodes are more likely to be pushed apart. For homophilic graphs, the optimal value of ε tends to be small, around 0.2, while heterophilic graphs show large values around 0.8 or 1.0.

FLOPs and memory usage. Table 4 compares the computational requirements of EvolveODE with DCRNN and STGCN. EvolveODE requires substantially fewer FLOPs than both baselines, although its runtime increases with hierarchy depth due to the sequential solution of the attraction–repulsion dynamics. GPU memory remains nearly constant across EvolveODE variants, increasing only from 7.53 GB to 7.58 GB, while RAM usage grows moderately with depth. Overall, EvolveODE provides a comparatively low computational cost while maintaining manageable memory requirements.

Sensitivity on integration length and step size. We show in Figure 7 the effect of changing solver step size and sorting termination time. Increasing both the integration time and the termination time jointly enables accurate sorting, namely by allowing the system to converge while keeping the approximated trajectory close to the real one. Larger step sizes lead to numerical instability, indicating incomplete sorting of variables into sub-processes.

Coupling Visualization. Figure 8 visualizes the coupling within a two-equation EvolveODE system on the Max Planck weather dataset. The figure illustrates higher-order interactions between the discovered sub-processes and observed variables. EvolveODE identifies three main sub-processes: the first consists of node indices, the second consists of nodes, and the third comprises interactions among nodes. The first group captures how temperature and atmospheric moisture jointly govern condensation and rainfall. The second captures the effects of strong winds on evaporation and atmospheric moisture transport by moving moisture away from the surface. The third represents transitions between rainy, cloudy, and sunny conditions through latent interactions among radiation, wind, vapor pressure, and air density. EvolveODE also learns latent inter-scale couplings, such as the connections between, which further link temperature and moisture dynamics.

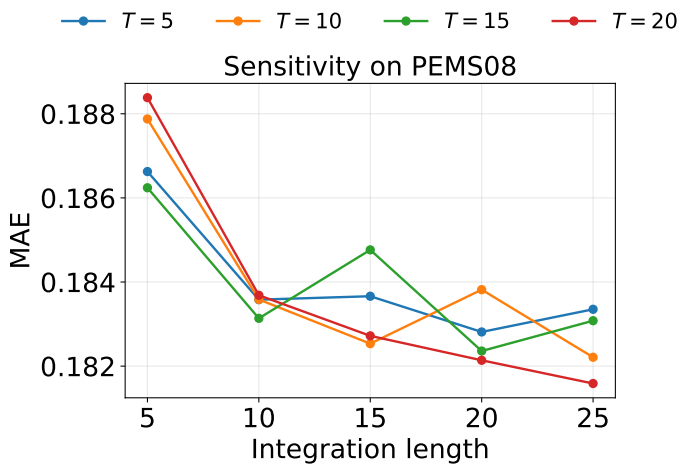


Figure 7: Sensitivity analysis on PEMS08 when varying the end-integration time T and step size. Shown is a two-equation set EvolveODE. Smaller step sizes require more integration steps.

Table 5: A standard HGNN, showcasing a saturation boundary.

Model	PEMS03		PEMS08	
	MAE	RMSE	MAE	RMSE
HGNN(1)	0.151	0.222	0.158	0.234
HGNN(2)	0.139	**0.210**	**0.145**	**0.217**
HGNN(3)	**0.138**	0.218	0.149	0.228

Saturation Boundary and EvolveODE. We verify the existence of a saturation boundary for a standard HGNNs. We give our results in Table 5.

Alternative to m-Cluster Potential Energy. We evaluate the efficacy of the m-well attraction–repulsion ODE against alternative clustering approaches. We denote the variant that uses DiffPool to identify sub-processes before integrating the m-well ODE as DiffPool+ODE. We further compare against the ACMP ODE, denoted ACMP-1 and ACMP-2 for zero and one additional set of equations from applying the clustering kernel, respectively. Tables 6 and 7 report the results of these comparisons.

For ACMP-1, performance remains competitive when no additional equations are introduced. However, performance deteriorates as the number of equations increases. This result supports the central motivation of EvolveODE: because the ACMP potential admits only two minima, the converged node features are restricted to two states, limiting the diversity of the resulting sub-process graphs. Consequently, these graphs cannot represent increasingly intricate interactions between sub-processes, leading to degraded performance as the system hierarchy expands.

For DiffPool+ODE, the results demonstrate that explicitly identifying sub-processes provides a tangible improvement in overall performance. More importantly, they show that the clustering mechanism alone does not account for the gains achieved by EvolveODE. Instead, the m-well potential provides the primary improvement by allowing the ODE to discover a richer set of sub-processes than the clustering kernel alone.

Coupling type sensitivity. In Table 8 we show the differences in coupling types, namely via connecting sub-process graphs in a cycle, using a pyramid structure from HGNNs, or connecting all sub-process graphs in a complete graph. We show that for smaller systems, specifically 2 or 3 sets of equations, the performance is largely the same. However, for larger sets, Cycle and Complete are the best coupling types. This shows that, through its

Table 6: Ablations of EvolveODE on traffic forecasting benchmarks.

Model	PEMS03		PEMS08	
	MAE	RMSE	MAE	RMSE
DiffPool+ODE	0.3208	0.4655	0.3059	0.4285
ACMP-1	0.3131	0.4590	0.2875	0.4074
ACMP-2	0.3440	0.4900	0.3646	0.5056
EvolveODE	**0.2620**	**0.3950**	**0.2410**	**0.3540**

Table 7: Ablations of EvolveODE on node classification benchmarks.

Model	Cora	Wisconsin
	Accuracy (%)	Accuracy (%)
DiffPool+ODE	31.90 ± 0.14	82.35 ± 2.65
ACMP-1	83.02 ± 0.33	60.00 ± 3.64
ACMP-2	32.68 ± 13.28	50.59 ± 3.37
EvolveODE	**84.20 ± 0.20**	**83.80 ± 1.50**

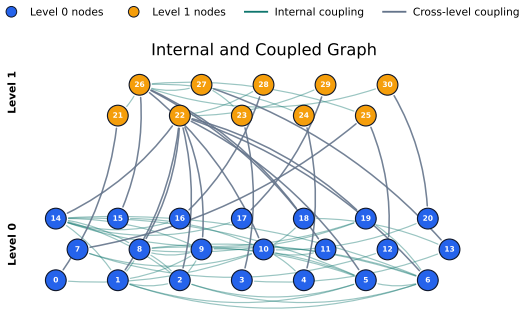


Figure 8: Coupling between the observational-level graph and the sub-process graph for a two-equation EvolveODE system on the Max Planck weather dataset.

Figure 9: Meteorological variables and their corresponding indices.

Index	Variable
0	Air pressure
1	Air temperature
2	Potential temperature
3	Dew-point temperature
4	Relative humidity
5	Saturation vapour pressure
6	Actual vapour pressure
7	Vapour-pressure deficit
8	Specific humidity
9	Water-vapour concentration
10	Air density
11	Mean wind velocity
12	Maximum wind velocity
13	Wind direction
14	Rainfall amount
15	Rainfall duration
16	Shortwave downward radiation
17	Photosynthetically active radiation
18	Maximum photosynthetically active radiation
19	Logger temperature
20	Ambient CO_2 concentration

continuous-time dynamics, EvolveODE allows the re-injection of coarse processes without relying on a hierarchical structure.

Long-horizon weather forecasting. We give the results of the long-horizon weather forecasting experiments in Table 9. At the 10 and 15-day mark, EvolveODE underperforms the baselines, which show strong medium-horizon capabilities. Yet, as lead time extends, the performance of EvolveODE remains stable relative to other models in the table. EvolveODEs' particle physics kernel prevents the blow-up of small errors common in an autoregressive setting, often occurring before forecast collapse. In contrast, the many nonlinear layers of transformers tend to exacerbate such errors. We further attribute this large improvement in stability to the direct emulation of the underlying weather dynamics, which further enforces high-fidelity rollouts as a consequence, by the 20-day mark, enforces high-fidelity rollouts. As a consequence, the ordering shifts, with EvolveODE now achieving the lowest error metrics, while the performance of baseline methods continues to deteriorate rapidly. By the 30-day mark, EvolveODE's gains are very substantial, 18.6% and 17.8% for MAE and RMSE, respectively.

Table 8: Topology sensitivity on PEMS08 with horizon and input buffer size set to 12.

Topology	2-Equations		3-Equations		4-Equations		5-Equations	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Cycle	0.1721	0.2570	0.1715	**0.2561**	**0.1716**	**0.2560**	**0.1711**	**0.2566**
Pyramid	**0.1718**	**0.2568**	**0.1714**	0.2567	0.1731	0.2587	0.1726	0.2571
Complete	0.1725	0.2575	0.1722	0.2568	0.1719	0.2572	0.1722	0.2567
Gains	**+0.2%**	**+0.1%**	**+0.1%**	**+0.2%**	**+0.2%**	**+0.4%**	**+0.6%**	**+0.1%**

Table 9: Long-horizon forecasting experimental results on the Max Planck weather dataset from 2025 to 2026. All results are averages of metrics over a particular day.

Model	Lead Time									
	10-days		15-days		20-days		25-days		30-days	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
TimeMixer++ (5)	0.477	0.658	0.471	0.601	0.476	0.601	0.804	1.123	0.915	1.141
iTransformer (48)	0.686	0.864	0.634	0.790	0.772	0.895	0.971	1.267	0.914	1.147
TimerXL (47)	0.620	0.816	0.575	0.758	0.633	0.788	0.844	1.152	0.756	1.020
EvolveODE	0.752	0.972	0.602	0.788	**0.582**	**0.777**	**0.750**	**1.057**	**0.638**	**0.865**
Gain (%)	**-17.6%**	**-16.0%**	**-4.2%**	**-3.7%**	**+8.8%**	**+1.5%**	**+12.7%**	**+9.0%**	**+18.6%**	**+17.8%**

Table 10: MAE across forecast horizons and input buffers.

Model		PEMS03				PEMS04				PEMS08			
		3	6	9	12	3	6	9	12	3	6	9	12
TimeMixer++ (5)	12	0.145	**0.153**	**0.161**	**0.168**	0.160	0.177	0.184	0.195	0.148	0.154	0.163	0.170
	24	0.142	0.153	0.165	0.173	0.160	0.172	0.180	0.183	0.147	0.159	0.171	0.172
	48	0.142	0.153	0.163	0.172	0.160	0.168	0.176	0.182	0.146	0.157	0.163	0.175
TimerXL (47)	12	0.144	0.159	0.173	0.187	0.165	0.181	0.198	0.216	0.152	0.167	0.183	0.198
	24	0.143	0.158	0.170	0.185	0.164	0.178	0.193	0.208	0.150	0.164	0.176	0.187
	48	0.144	0.157	0.169	0.180	0.165	0.179	0.192	0.205	0.150	0.163	0.176	0.188
iTransformer (48)	12	0.145	0.161	0.178	0.192	0.165	0.180	0.199	0.217	0.153	0.170	0.186	0.203
	24	0.141	0.155	0.167	0.179	0.162	0.176	0.190	0.204	0.145	0.157	0.168	0.179
	48	0.141	0.153	0.163	0.174	0.161	0.174	0.186	0.198	0.145	0.155	0.165	0.174
FEDformer (39)	12	**0.140**	0.156	0.166	0.173	0.159	0.173	0.180	0.185	0.153	0.164	0.174	0.181
	24	0.151	0.162	0.170	0.176	0.172	0.176	0.184	0.188	0.158	0.170	0.178	0.184
	48	0.151	0.162	0.170	0.176	0.172	0.173	0.184	0.188	0.159	0.170	0.178	0.184
NSTransformer (55)	12	0.143	0.154	0.166	0.172	0.162	0.169	0.178	0.186	0.153	0.162	0.172	0.181
	24	0.149	0.161	0.174	0.178	0.165	0.173	0.179	0.183	0.153	0.163	0.169	0.180
	48	0.163	0.171	0.181	0.192	0.172	0.179	0.184	0.189	0.169	0.169	0.176	0.187
MAGE	12	0.143	**0.153**	0.162	**0.168**	0.161	0.170	0.177	0.182	0.148	0.156	0.162	0.170
	24	0.143	0.154	0.162	0.170	0.161	0.171	0.177	**0.180**	0.146	0.155	0.162	**0.166**
	48	0.144	0.154	0.164	0.170	0.161	0.172	0.178	0.184	0.148	0.156	0.164	0.169
CITRUS	12	0.152	0.163	0.178	0.182	0.172	0.175	0.205	0.195	0.151	0.161	0.173	0.178
	24	0.148	0.160	0.174	0.180	0.163	0.173	0.180	0.188	0.152	0.163	0.172	0.181
	48	0.148	0.165	0.172	0.179	0.162	0.170	0.178	0.184	0.154	0.168	0.173	0.179
EvolveODE	12	**0.140**	**0.153**	0.163	0.172	**0.155**	**0.164**	**0.174**	**0.181**	**0.140**	**0.153**	**0.161**	**0.168**
	24	**0.139**	**0.151**	**0.160**	**0.169**	**0.154**	**0.163**	**0.172**	0.181	**0.141**	**0.153**	**0.160**	0.171
	48	**0.139**	**0.149**	**0.158**	**0.165**	**0.153**	**0.165**	**0.170**	**0.180**	**0.141**	**0.153**	**0.159**	**0.165**
Gain (%)	12	**+0.0%**	**+0.0%**	**-1.2%**	**-2.4%**	**+2.5%**	**+3.0%**	**+1.7%**	**+0.5%**	**+5.4%**	**+0.6%**	**+0.6%**	**+1.2%**
	24	**+1.4%**	**+1.3%**	**+1.2%**	**+0.6%**	**+3.8%**	**+4.7%**	**+2.8%**	**-0.6%**	**+2.8%**	**+1.3%**	**+1.2%**	**-3.0%**
	48	**+1.4%**	**+2.6%**	**+3.1%**	**+2.9%**	**+4.4%**	**+1.8%**	**+3.4%**	**+1.1%**	**+2.8%**	**+1.3%**	**+2.5%**	**+2.4%**