FIELD NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic interactions are modeled in many ways, from temporal graphs and recurrent state updates to Neural ODEs and physics-based dynamical systems. Inspired by classical field theory, we instead model interactions as the evolution of a latent field. We introduce *Field Neural Networks* (FNNs), which parameterize simple neural updates with field-equation structure to capture diffusion-like and wave-like dynamics. On controlled simulations, FNNs recover the underlying dynamics across various hidden synthetic topologies and remain robust under long rollouts and sparse observations. On real temporal link-prediction benchmarks, they remain competitive with established temporal graph models while using roughly an order of magnitude fewer parameters. Mechanistic ablations further reveal dataset-dependent reliance on learned dynamical coefficients and velocity state. These results suggest that field equations can serve as compact inductive biases for interaction data, providing a structured way to model how activity enters, propagates through, and persists within relational systems.

1 INTRODUCTION

Communication logs, face-to-face contact records, and transaction histories describe systems of entities that interact over time. These records reveal who interacted with whom and when, and temporal graph models use them to learn interaction patterns and predict future events (Trivedi et al., 2019; Kumar et al., 2019; Xu et al., 2020; Rossi et al., 2020). However, observed events do not directly reveal all of the evolving states relevant to future behavior. An email log, for example, records exchanges without directly measuring how information or activity carried by its participants changes over time. The resulting challenge is to infer from observable interactions how latent states evolve and influence future interactions.

One way to structure this hidden evolution is to draw on statistical physics and network science, which describe collective behavior through local interactions and evolving states, including diffusion, social influence, and spreading processes (Castellano et al., 2009; Vespignani, 2012). Their use here does not imply that arbitrary interaction data obey a known physical law. Rather, they provide a dynamical vocabulary for describing how activity enters a system, spreads through it, dissipates, and persists.

We adopt a *field* representation to express this vocabulary. A physical field assigns a quantity, such as temperature or pressure, to each spatial location at a given time (Nastase, 2019; Strauss, 2008). Here, each entity is assigned a latent vector, and these vectors collectively form a field over a relational domain. Field equations provide a template for structuring how this representation evolves in response to observed interactions.

We introduce *Field Neural Networks* (FNNs) for this setting. An FNN structures latent evolution through a small set of mechanisms inspired by field dynamics: event forcing, damping, restoration, relational propagation, and, in the second-order model, inertia. Interactions determine where activity enters the field and how entities are related, while an ordinary predictive loss determines which latent dynamics are useful for the task. As illustrated in Fig. 1, an FNN begins only with observed interactions and learns both the latent field and the structured dynamics governing its evolution.

This design builds on a broader use of physical structure to guide learned dynamics. Physics-informed neural networks and physics-informed neural operators incorporate known differential equations into training (Raissi et al., 2019; Li et al., 2024), while Hamiltonian and Lagrangian neural networks derive dynamics from learned energy functions (Greydanus et al., 2019; Cranmer et al.,

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.

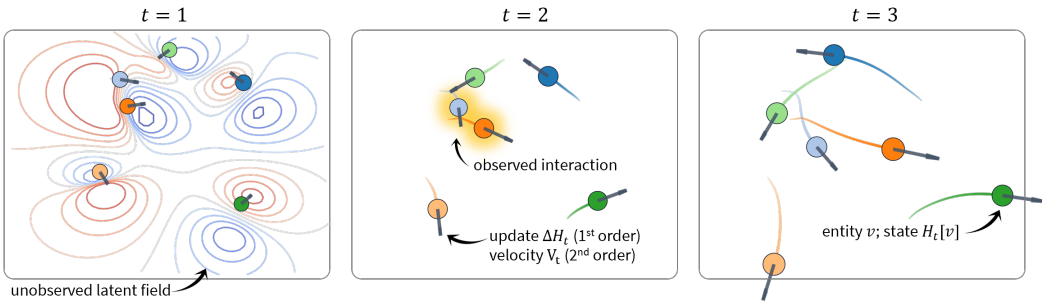

Figure 1: From observed interactions, a Field Neural Network learns a latent field and the structured dynamics governing its evolution. Each particle represents an entity with a learned state and field vector.

2020). Related approaches learn latent dynamics from spatiotemporal measurements (Regazzoni et al., 2024), and graph models use diffusion or coupled-oscillator dynamics to structure representation learning (Chamberlain et al., 2021; Rusch et al., 2022).

In FNNs, the field itself is latent: entities define its domain, interactions drive its evolution, and prediction provides supervision. Field equations therefore serve not as physical laws to be recovered, but as inductive biases for learning latent dynamics over interaction streams.

This structure makes an FNN both predictive and mechanistically structured. We develop first- and second-order FNNs, with the latter introducing a velocity-like state to represent inertia. Both variants construct latent states from interaction history and learn their trainable components through downstream prediction losses.

We evaluate FNNs on controlled simulations and real temporal link-prediction benchmarks. Controlled diffusion and wave experiments test prediction across five hidden topologies, rollout horizons of up to 20 steps, and varying levels of event retention, while dedicated experiments assess topology and parameter recovery. Across nine real interaction benchmarks, FNNs offer a competitive accuracy–parameter trade-off on several datasets. Component ablations reveal dataset-dependent benefits from learning dynamical coefficients and using velocity features, allowing us to examine which parts of the structured update support prediction. These results support field equations as compact inductive biases for learning predictive representations from interaction histories.

2 Field Neural Networks

Field Neural Networks combine the event interface of temporal graph models with the mechanistic structure of field equations. Events introduce activity into the field and define the relational domain over which that activity may propagate. The latent field is trained end to end through its usefulness for predicting downstream node, edge, or interaction outcomes.

In its general formulation, an FNN organizes these quantities into three components. First, it maintains a latent field $H_t \in \mathbb{R}^{N\times q}$, where each row assigns a q-dimensional state to one of the N entities. Second, events observed during interval t produce a node-wise forcing field $F_t \in \mathbb{R}^{N\times q}$, representing activity entering the system. Third, a relational operator $L_t \in \mathbb{R}^{N\times N}$ represents coupling between entities. An FNN uses F_t and L_t to evolve the latent field from H_t to H_{t+1}. After the field is updated, predictions may be read out from the latent field and velocity states, or from learned relational parameters.

2.1 Field Theory

Classical mechanics describes a system using a finite collection of generalized coordinates, such as the positions of a set of particles. Field theory extends this description by assigning a quantity to every location in a domain. Let $\Omega \subseteq \mathbb{R}^d$ denote that domain. A classical field is a function $\phi : \Omega \times [t_0, t_1] \to \mathbb{R}^{d_\phi}$, where $\phi(\mathbf{x}, t)$ gives the field state at location $\mathbf{x}$ and time t.

Field equations describe how these local states evolve together. A broad class of driven and damped field dynamics can be written as

$$\underbrace{\rho\,\partial_{tt}\phi}_{\text{inertia}} + \underbrace{\gamma\,\partial_t\phi}_{\text{damping}} - \underbrace{\kappa\,\nabla^2\phi}_{\text{spatial coupling}} + \underbrace{\lambda\phi}_{\text{restoring force}} = \underbrace{f(\mathbf{x},t)}_{\text{external forcing}} . \tag{1}$$

Equation 1 is a general template rather than a universal physical law. Inertia allows changes in the field to continue through time, while damping causes that motion to dissipate. The spatial Laplacian $\nabla^2\phi$ compares the field at each location with its surroundings, allowing disturbances to spread or relax across the domain. The restoring term draws the field toward a baseline, while forcing introduces activity from outside the current field state. The coefficients ρ, γ, κ, and λ control the relative strength of these mechanisms. Appendix A derives this equation from a quadratic Lagrangian density and provides a more detailed explanation of its components.

The physical interpretation above assumes a continuous spatial domain. Interaction data need not provide such a space. Instead, we take the domain to be a finite set of entities whose relationships are revealed through events. An FNN treats each entity as a location in a discrete field. The continuous field $\phi(\mathbf{x},t)$ is replaced by a latent node state $H_t[v] \in \mathbb{R}^q$, and the complete distributed field is represented by $H_t \in \mathbb{R}^{N\times q}$.

Events connect the classical field equation to the predictive model in two ways. First, events are aggregated at entities to produce a forcing field F_t, the discrete analogue of $f(\mathbf{x},t)$. Second, observed interactions can be used to construct a graph operator $L(t)$, which replaces the continuous spatial Laplacian and determines how latent activity can propagate among entities. The resulting correspondence is

$$\phi(\mathbf{x},t) \longrightarrow H(t), \qquad f(\mathbf{x},t) \longrightarrow F(t), \qquad -\nabla^2\phi \longrightarrow L(t)H(t). \tag{2}$$

Applying Equation 2 to Equation 1 gives the relational field equation

$$\rho\,\partial_{tt}H(t) + \gamma\,\partial_t H(t) + \kappa\,L(t)H(t) + \lambda H(t) = F(t). \tag{3}$$

Crucially, $H(t)$ is not a measured physical field. It is an internal representation constructed from interaction history, with no direct supervision of its values or derivatives. Instead, downstream prediction errors train the components that construct and evolve the field. As illustrated in Figure 1, the field equations constrain how this representation evolves, while the prediction task determines which representations are useful.

Equation 3 provides the continuous-time template motivating FNNs. After partitioning the event stream into discrete intervals, we write H_t, F_t, and L_t for the corresponding quantities at interval t.

2.2 Event-Induced Fields

The relational field in Equation 3 requires a forcing field F_t and a coupling operator L_t. Interactions can inform both: event attributes specify incoming activity, while source–destination pairs identify connections through which activity may propagate.

Let $e_i = (s_i, d_i, x_i, \tau_i)$ denote an interaction event, where s_i and d_i are its source and destination entities, x_i contains optional event features, and τ_i is its timestamp. We write $\mathcal{E}_t$ for the events processed during interval t. Events may be grouped according to their timestamps or processed individually.

Persistent connection weights. For each modeled connection, the FNN maintains a learned logit θ_{ij} and corresponding gate

$$g_{ij} = \sigma(\theta_{ij}). \tag{4}$$

The gate weights the contribution of an observed event to its destination. These connection weights persist across intervals and are optimized during training; they are not reconstructed independently from each interval's event adjacency.

Event forcing. Let $b_i \in \mathbb{R}^q$ denote the input associated with event e_i. The connection gate in Eq. 4 modulates this input:

$$m_i = g_{s_i d_i}\, b_i. \tag{5}$$

We aggregate these drives at their destinations:

$$F_t[v] = \sum_{\substack{i:e_i \in \mathcal{E}_t \\ d_i = v}} m_i. \tag{6}$$

Thus, $F_t[v]$ represents incoming activity assigned to entity v.

Relational propagation. In variants with input propagation, a sparse K-hop operator distributes the destination-aggregated forcing over the training graph before the field update. Setting $K = 0$ retains destination-only forcing.

Explicit coupling of the existing latent field is a different operation. In the general formulation, L_t is a graph analogue of the negative spatial Laplacian in Equation 1. When chosen as a symmetric normalized graph Laplacian, the term $-\kappa L_t H_t$ promotes degree-normalized agreement across connected entities at all time steps.

2.3 Field-Structured State Updates

Equation 3 describes continuous-time relational field dynamics. Interaction data, however, are processed at discrete event timestamps or intervals. We therefore derive discrete update laws that map H_t to H_{t+1}. The first-order FNN models diffusion and relaxation, while the second-order FNN additionally maintains a velocity-like state to represent inertia.

2.3.1 First-Order Field Neural Network

The first-order FNN assumes that the latent field has no independent momentum: its rate of change is determined only by its current state, relational coupling, and event forcing. Accordingly, setting $\rho = 0$ in Eq. 3 yields

$$\gamma\, \partial_t H(t) + \kappa L(t) H(t) + \lambda H(t) = F(t). \tag{7}$$

Assuming $\gamma > 0$, divide Eq. 7 by γ, define $\delta = \lambda/\gamma$ and, for notational simplicity, we reuse κ and F_t for κ/γ and F_t/γ, respectively. This yields

$$\partial_t H(t) = -\delta H(t) - \kappa L(t) H(t) + F(t). \tag{8}$$

Here δ is the first-order state-decay coefficient; it is distinct from the velocity-damping coefficient γ used later. Applying forward Euler to Equation 8 over an interval Δt yields

$$H_{t+1} = H_t + \Delta t \left(-\delta H_t - \kappa L_t H_t + F_t\right). \tag{9}$$

Thus Equation 9 is the discrete first-order FNN update. Appendix B provides the full derivation from Equation 3.

2.3.2 Second-Order Field Neural Network

The second-order FNN retains the inertial term in Eq. 3. Defining the velocity-like latent field $V(t) = \partial_t H(t)$ rewrites this equation as two coupled first-order equations. Assuming $\rho > 0$, we divide by ρ and reuse γ, κ, and $F(t)$ for γ/ρ, κ/ρ, and $F(t)/\rho$, respectively. For $\lambda \geq 0$, we write $\omega^2 = \lambda/\rho$. This yields

$$\partial_t V(t) = -\gamma V(t) - \kappa L(t) H(t) - \omega^2 H(t) + F(t), \tag{10}$$

$$\partial_t H(t) = V(t), \tag{11}$$

Equation 10 shows explicitly how the second-order model extends Equation 8: $V(t)$ introduces inertia, while γ, κ, ω^2, and $F(t)$ govern velocity damping, relational coupling, local harmonic restoration, and external forcing, respectively.

Applying forward Euler to Eq. 10, then using the new velocity in Eq. 11, gives

$$V_{t+1} = V_t - \Delta t \left(\gamma V_t + \kappa L_t H_t + \omega^2 H_t\right) + \Delta t F_t, \tag{12}$$

$$H_{t+1} = H_t + \Delta t V_{t+1}. \tag{13}$$

Equation 12 carries the previous velocity forward with Euler coefficient one and dampens it through $(1 - \gamma\Delta t)V_t$. The $\omega^2 H_t$ and $\kappa L_t H_t$ terms respectively supply local restoration and relational coupling. Because Equation 13 uses the newly computed V_{t+1}, it is a semi-implicit, symplectic-Euler-style position update. Appendix C derives Equations 10–13 from Equation 3.

2.4 The End-to-End Field Neural Network

An FNN combines persistent connection weights, an evolving latent field, and task-specific prediction heads. The field $H_t \in \mathbb{R}^{N\times q}$ and, for second-order models, velocity $V_t \in \mathbb{R}^{N\times q}$ summarize the observed interaction history. These states evolve as events arrive; they are not independently optimized node embeddings.

Field evolution. Destination aggregation produces the forcing field F_t through Eq. 6. Where enabled, the K-hop propagator distributes this input over neighboring entities before the state update. The resulting forcing advances the field according to the first-order formulation in Eq. 9 or the second-order formulation in Eqs. 12 and 13.

The general formulation permits explicit coupling through $L_t H_t$. In the configurations evaluated here, however, we propagate incoming event activity rather than applying L_t directly to the stored latent field at every step. In the second-order case, the velocity is updated first, and the newly computed velocity then advances the field. The resulting states are retained for subsequent predictions and updates.

State-based prediction. A decoder maps the latent states to the output required by the task. Define

$$Z_t[i] = \begin{cases} H_t[i], & \text{first order,} \\ [H_t[i], V_t[i]], & \text{second order.} \end{cases} \tag{14}$$

After processing interval t, a candidate interaction between u and v can be scored using

$$\widehat{y}_{uv,t+1} = \mathrm{Dec}_{\theta_D}\left(Z_{t+1}[u], Z_{t+1}[v]\right). \tag{15}$$

The target interaction is not incorporated into the state before it is predicted.

For synthetic force prediction, the decoder is a two-layer MLP that maps the concatenated endpoint states to a force vector $\widehat{\mathbf{f}}_{u\to v,t+1}$. For real-world link prediction, a shared learned projection first maps each endpoint state into the DyGLib evaluation harness (Yu et al., 2023); its link decoder then produces an interaction score.

Learning from downstream outcomes. The trainable components are optimized using the relevant task loss:

$$\min_{\Theta} \sum_t \mathcal{L}_{\mathrm{task}}\left(\widehat{Y}_{t+1}, Y_{t+1}\right), \tag{16}$$

where Θ contains the parameters enabled for learning in the chosen experiment.

For state-based outputs, gradients pass through the decoder and differentiable field updates to the trainable dynamics and input weights. The field and velocity themselves have no supervision targets: training shapes the components that construct and evolve them through their usefulness for downstream prediction.

3 Experimental Methodology

Our experiments ask when an event stream is better modeled as the evolution of a latent field than as an unconstrained recurrent memory. We first test whether FNNs can recover known field dynamics

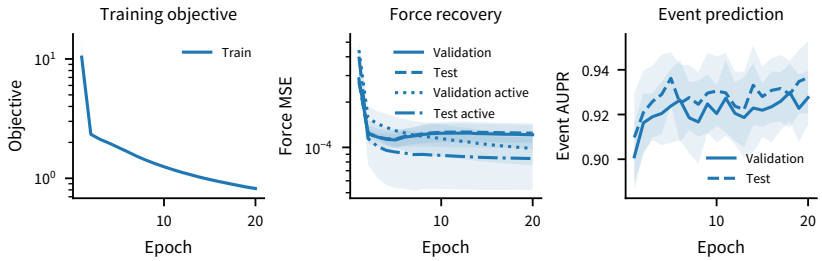

Figure 2: Learning behavior on second-order wave dynamics over a ring topology. Curves show the mean across five random seeds, with error bands indicating standard error. Force-prediction error decreases rapidly and stabilizes alongside the downstream ranking and event-detection metrics.

from controlled simulations, then ask whether the same inductive bias is useful on real interaction data. Each dataset is represented as an ordered event stream, $\mathcal{E}_{1:T}$.

Synthetic simulations provide known topology and dynamical coefficients, but these are used only to generate data and evaluate recovery. Except in designated ablations, the model observes only the resulting interactions, not the topology, field, or generator parameters.

When the FNN is initialized with the same physical coefficients (γ, κ, etc.) used by the simulator, it converges rapidly to a near-perfect predictor because its update rule is designed to represent those dynamics directly. To make this diagnostic more demanding, we instead initialize the FNN with random coefficients and hold them fixed during training. The model must therefore improve by learning a useful latent field despite misspecified dynamics, rather than by simply matching the simulator parameters. Later experiments relax this constraint and learn the latent field and dynamical coefficients jointly.

Because retaining every simulated interaction can produce a dense stream, we treat an interaction as active when its force falls in the top quartile of forces observed in the training set. This emphasizes interactions carrying the dominant dynamics, such as the advancing front of a propagating wave, rather than small residual forces.

We compare FNN against GRU (Cho et al., 2014), DeepSets (Zaheer et al., 2017), SetTransformer (Lee et al., 2019), modern Hopfield (Ramsauer et al., 2021), Lagrangian (Cranmer et al., 2020), and Hamiltonian (Greydanus et al., 2019) models. On real-world datasets, we additionally compare against temporal graph models including JODIE (Kumar et al., 2019), TGN (Rossi et al., 2020), TGAT (Xu et al., 2020), GraphMixer (Cong et al., 2023), and DyGFormer (Yu et al., 2023).

We report force error for event-feature prediction, MRR and Hits@k for next-destination ranking, and AUROC/AUPR/AP for event detection. Rollout experiments feed each prediction back into the model to generate subsequent predictions. All splits are chronological, model selection uses validation loss, AP, and AUCROC, and unless otherwise specified, results report the mean and standard error over five random seeds. Additional details appear in Appendix D.

3.1 Learning curve

We begin with a small diagnostic to establish that the FNN training objective produces stable predictive behavior before moving to the broader benchmark. We simulate second-order wave dynamics on a ring topology, with an external impulse applied every twelve time steps. At each step, the simulator produces the interaction events induced by the dynamics in Eq. 13.

Models are trained to predict the force associated with interactions in the next time step. We evaluate the same predictions in two additional ways: MRR ranks each active interaction against ten randomly sampled inactive destinations, while event detection compares active interactions with sampled inactive pairs using AUPRC.

Figure 2 shows rapid improvement across these complementary views of the task. Training loss and force-prediction error decrease sharply during the first several epochs and stabilize after roughly ten epochs; event-detection performance stabilize over the same interval. The agreement across

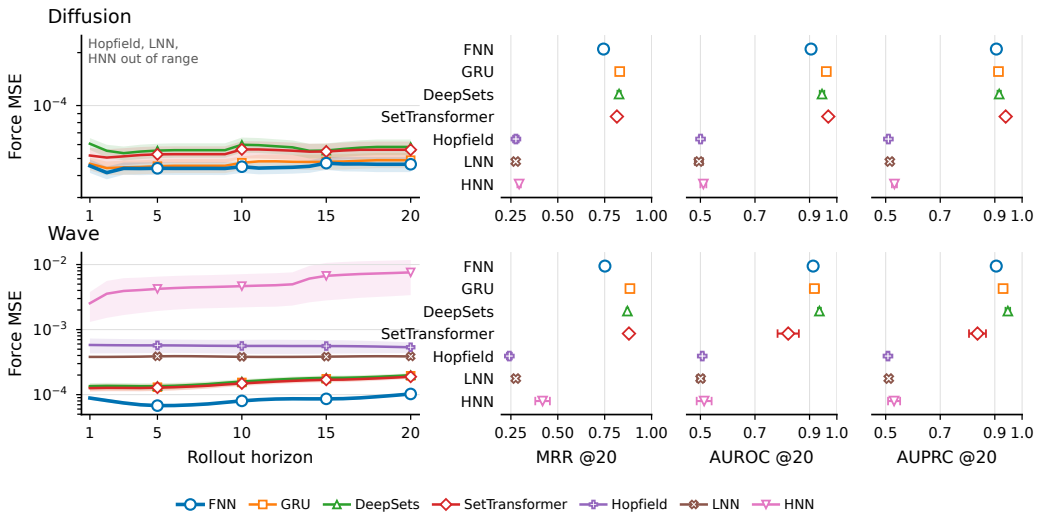

Figure 3: Predictive performance across diffusion and wave dynamics. Left panels show force-prediction error as models are rolled forward for up to 20 steps, aggregated across the five hidden interaction topologies. Points and error bars in the remaining panels report model performance at rollout horizon 20 for mean reciprocal rank (MRR), event AUROC, and event AUPRC. Curves and points summarize performance across topologies and random seeds.

objectives indicates that the model is not merely fitting force magnitudes, but is learning a latent field that consistently identifies where interactions will occur and how strongly they will act.

3.2 Dynamics and Topology Simulations

We next evaluate predictive performance across dynamics and topologies. We generate first-order diffusion dynamics using Eq. 9 and second-order wave dynamics using Eq. 13. Each process is simulated over five hidden topologies: a ring, grid, torus, a grid interrupted by narrow apertures (double slit), and a grid containing multiple obstacles (Swiss cheese) (Fig A1). Together, these settings test whether the field-based inductive bias remains useful as both the governing dynamics and the geometry through which interactions propagate change.

For the dynamics-topology benchmark experiment, the physical coefficients are held fixed during training so that performance differences primarily reflect how well each model learns and propagates the underlying field dynamics, rather than its ability to recover the generating coefficients.

Figure 3 summarizes predictive performance, averaged across the five topologies and random seeds, separately for diffusion and wave dynamics. On this aggregate comparison, FNNs achieve the lowest force-prediction error and maintain their advantage as the rollout horizon increases. The separation is particularly pronounced for wave dynamics. At horizon 20, FNNs also achieve the strongest aggregate ranking and event-discrimination performance. These results suggest that the field representation preserves interaction-relevant information over extended rollouts. Figures A2 and A3 report the corresponding results for each dynamics–topology combination, showing the variation obscured by the aggregate summary.

Training-horizon Next, we separate the horizon optimized during training from the horizon used at test time. While all runs are evaluated with a fixed 20-step rollout, they are only trained with a next-step training $K_{\text{train}} = 1$. We extend K_{train} to longer rollouts and compare performance for FNN and benchmark models. Results, shown in Figs A4 and A5, show that longer training rollouts do not change the FNN results substantially.

Observation Threshold Recall, by default, that we only consider an interaction active if it is in the top quartile of activation edges. We vary that signal-threshold for each simulation and topology.

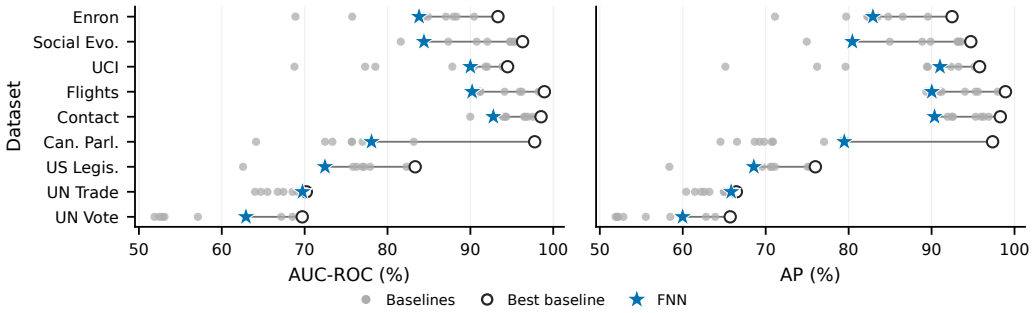

Figure 4: Real-world link-prediction performance under random negative sampling. Stars identify FNN, gray points show the comparison models, and open circles mark the strongest baseline for each dataset and metric. Higher AUROC and AP are better.

We select dense, medium, sparse, and very-sparse regimes by retaining the top 5, 20, 50, and 100% of interactions, respectively. This tests how each model responds when the interactions generated by the field are only partially observed. Results, shown in Figs A6 and A7, shows that FNN is robust to the percentage of observed events.

Learning Physical Parameters To this point, all experiments held the FNN physical coefficients (*e.g.*, Δt, γ, ω) fixed at values intentionally different from those used by the synthetic generators. We now ask whether the complete FNN can jointly recover both the relational topology and the physical dynamics that produced the data.

When the physical parameters and relational operator are optimized jointly from the start, different coefficients can compensate for one another and converge to incorrect values while still producing accurate predictions (Appendix E.1). We therefore consider two training strategies: direct joint optimization and an alternating block-coordinate schedule that separately updates the relational operator and physical parameters. Full training details and comparisons of prediction, topology recovery, and parameter recovery are provided in Appendix E.1 and Figs. A9–A11.

3.3 Real-World Datasets

We next evaluate FNN on real interaction streams using DyGLib, a general framework for temporal graph learning (Yu et al., 2023). We integrate FNN into its native link-prediction pipeline, use their chronological data splits and negative sampling settings, and perform a small hyperparameter search with ASHA (Li et al., 2020). Over nine datasets we report transductive test AUROC and average precision (AP), compared against the state of the art benchmarks. FNN results summarize five random seeds, with error bars showing ± 1 SEM; baseline scores are reproduced from published results (Yu et al., 2023).

Figure 4 shows that FNN is competitive with several established temporal graph models, although the strongest baselines retain advantages on some datasets. These results demonstrate that a compact, field-structured state update can provide useful predictions beyond the simulated systems that motivate its design. Its contribution is therefore not uniform superiority in predictive accuracy, but a complementary accuracy–parameter trade-off, examined next.

Figure 5 illustrates this trade-off on three selected datasets (see Fig. A12 for the full result set). FNN's parameter count depends on the number of learned connections in its relational operator, and remains below 100,000 on these graphs—roughly an order of magnitude smaller than TGN's 964,000 parameters. On each dataset, FNN lies on the accuracy–parameter frontier among the included parameterized models. These results suggest that the field-based inductive bias provides a parameter-efficient foundation for temporal prediction: a compact dynamical model can achieve competitive performance without matching the capacity of larger architectures.

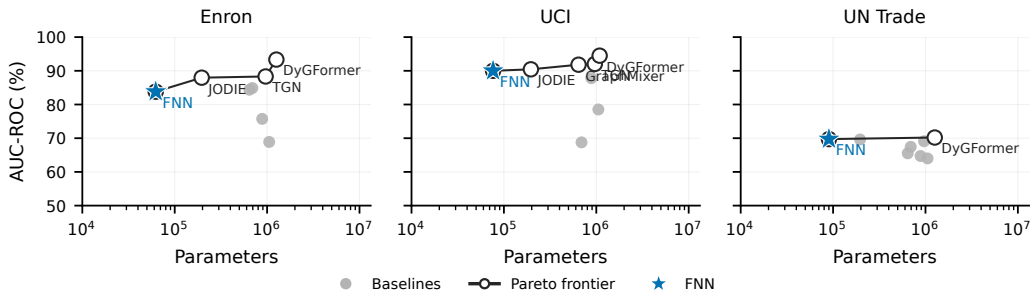

Figure 5: AUROC versus parameter count on Enron, UCI, and UN Trade. Stars identify FNN; gray points show comparison models with available parameter counts. The horizontal axis is logarithmic. Connected open circles indicate models for which no included alternative achieves equal or higher AUROC with fewer parameters, or higher AUROC with the same number of parameters.

4 Discussion

Field Neural Networks suggest that interactions can be modeled usefully through the language of fields, even when no physical field is directly observed. Across controlled simulations, the field-based inductive bias supports accurate prediction over different dynamics, topologies, rollout horizons, and levels of observation. On real interaction data, FNNs remain competitive with substantially larger temporal graph models while using far fewer parameters. The value of the approach is therefore not that field dynamics dominate every alternative, but that the inductive bias of this physics-inspired model with a small set of structured mechanisms—forcing, coupling, damping, and inertia—can explain a surprising amount of temporal interaction behavior.

Additionally, the real-world experiments show that these mechanisms are not utilized equally across datasets. Learned pairwise gates contribute little under the evaluated configuration, whereas adapting the dynamical coefficients can be important. Contact and UCI are especially sensitive to fixing the learned dynamics. Post-hoc perturbations of ω provide complementary results, as changing the restoring frequency has close to no effect on Can. Parl., but produces large, directionally different changes on UN Trade and UN Vote.

The velocity state provides another example of this dataset-dependent mechanism use. Separately trained first- and second-order models often achieve similar predictive performance, and neither order dominates across all datasets. However, hiding only the direct velocity contribution from a trained second-order readout causes substantial losses on the majority of datasets. A retrained model can adapt its representation when a mechanism is unavailable, whereas a post-hoc intervention reveals what an already trained predictor actually uses.

Taken together, these results suggest that FNN should not be interpreted as recovering one universal physical law from real interaction data. Instead, the architecture provides a bank of structured dynamical mechanisms, whose functional importance is dataset-dependent. This is the main interpretability advantage of the formulation: the mechanisms can be frozen, perturbed, or hidden individually, allowing their contribution to prediction to be tested directly.

Importantly, functional importance should be distinguished from physical identifiability. On real data, where no generating coefficients are available, learned γ, ω, and related parameters should therefore be viewed as task-adapted dynamical coefficients rather than literal physical constants. Their value can be found in exposing the temporal mechanisms used by the predictor, not in assigning a unique physical interpretation to every learned scalar.

This distinction points to the main opportunity for future work. FNNs currently impose a deliberately simple family of field dynamics; richer nonlinear fields, heterogeneous coefficients, and larger latent fields may improve predictive capacity without abandoning the structured inductive bias. More broadly, the results suggest that temporal interaction modeling need not choose between mechanistic structure and learned representations. Field equations can provide the structure, while downstream prediction determines the latent dynamics that are useful for the task.

AI USE STATEMENT

In this work, we used generative AI tools for implement methods, creating unit tests, and assisting with creating the plotting code. The paper was not written with an AI tool, but AI tools were used to provide feedback.

We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results.

We have reviewed all AI-assisted work. We manually checked unit tests and results for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The source code, results notebooks, and all scripts needed to generate and fully reproduce all figures and tables is available at https://anonymous.4open.science/r/fnn-955B.

REFERENCES

Claudio Castellano, Santo Fortunato, and Vittorio Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81(2):591–646, 2009. doi: 10.1103/RevModPhys.81.591.

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*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1407–1418. PMLR, 2021. URL https://proceedings.mlr.press/v139/chamberlain21a.html.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636*, 2023.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1269–1278. ACM, 2019. doi: 10.1145/3292500.3330895.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-Tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. *Proceedings of machine learning and systems*, 2:230–246, 2020.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Horatiu Nastase. *Classical Field Theory*. Cambridge University Press, 2019. doi: 10.1017/9781108569392.

Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021.

Francesco Regazzoni, Stefano Pagani, Matteo Salvador, Luca Dede', and Alfio Quarteroni. Learning the intrinsic dynamics of spatio-temporal processes through latent dynamics networks. *Nature Communications*, 15(1):1834, 2024.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

T. Konstantin Rusch, Benjamin P. Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18888–18909. PMLR, 2022. URL https://proceedings.mlr.press/v162/rusch22a.html.

Walter A. Strauss. *Partial Differential Equations: An Introduction*. John Wiley & Sons, 2 edition, 2008.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

Alessandro Vespignani. Modelling dynamical processes in complex socio-technical systems. *Nature Physics*, 8(1):32–39, 2012. doi: 10.1038/nphys2160.

Da Xu, Chuanwei Ruan, Evrim Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700, 2023.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

A From Classical Fields to Field Dynamics

This appendix develops the classical field equation used to motivate Field Neural Networks. The goal is not to provide a complete treatment of classical field theory, but to explain where the principal components of the FNN update come from.

A.1 From Particle Coordinates to Fields

Classical mechanics often describes a system using a finite collection of generalized coordinates. For example, the position of a particle moving along a line can be represented by a single function $q(t)$. A system of several particles can be represented by a vector

$$\mathbf{q}(t) = [q_1(t), q_2(t), \ldots, q_n(t)] .$$

Each coordinate changes with time, but the number of coordinates is finite.

A field extends this idea by assigning a value to every location in a domain. Let $\Omega \subseteq \mathbb{R}^d$ denote a spatial domain. A classical field is a function

$$\phi : \Omega \times [t_0, t_1] \to \mathbb{R}^{d_\phi},$$

where $\phi(\mathbf{x}, t)$ gives the field value at location $\mathbf{x} \in \Omega$ and time t. When $d_\phi = 1$, the field is scalar. Temperature is a familiar scalar field: every location has one temperature at each time. When $d_\phi > 1$, the field is vector-valued. A velocity field, for example, assigns both a speed and direction to every location in a moving fluid (Nastase, 2019; Strauss, 2008).

A field can therefore be understood as a very large collection of local states. The field equation specifies how all of those states change together.

A.2 Derivatives of a Field

Because a field varies through both space and time, its dynamics involve temporal and spatial derivatives.

The first temporal derivative,

$$\partial_t \phi = \frac{\partial \phi}{\partial t},$$

measures how quickly the field is changing at a fixed location. If $\phi(\mathbf{x}, t)$ represents the height of a water surface, then $\partial_t \phi$ is the vertical velocity of the surface at $\mathbf{x}$.

The second temporal derivative,

$$\partial_{tt} \phi = \frac{\partial^2 \phi}{\partial t^2},$$

measures how that velocity is changing. It is therefore analogous to acceleration in ordinary mechanics.

Spatial derivatives describe how the field differs across nearby locations. The gradient, $\nabla \phi$, points in the direction in which the field increases most rapidly. For a temperature field, the gradient points toward warmer locations.

The Laplacian is the divergence of the gradient:

$$\nabla^2 \phi = \nabla \cdot \nabla \phi.$$

Intuitively, the Laplacian compares the value at one location with the values around it. In one spatial dimension,

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2}.$$

Suppose a point on a string is higher than both of its neighbors. The field is curved downward at that location, and its Laplacian is negative. If the point is lower than its neighbors, the field is

curved upward, and its Laplacian is positive. This local comparison allows the Laplacian to describe diffusion, smoothing, and wave propagation.

A.3 The Lagrangian Density

In ordinary Lagrangian mechanics, the Lagrangian is typically the difference between kinetic and potential energy: $L = T - V$.

A field has energy distributed throughout a domain, so field theory replaces the ordinary Lagrangian with a *Lagrangian density*. The density describes the contribution of each local region to the system's dynamics.

Consider the quadratic Lagrangian density (Nastase, 2019)

$$\mathcal{L} = \underbrace{\frac{\rho}{2}\left\|\partial_t \phi\right\|^2}_{\text{kinetic contribution}} - \underbrace{\frac{\kappa}{2}\left\|\nabla \phi\right\|^2}_{\text{spatial coupling}} - \underbrace{\frac{\lambda}{2}\left\|\phi\right\|^2}_{\text{restoring potential}} . \tag{17}$$

The first term is analogous to kinetic energy. It is large when the field is changing rapidly. The parameter ρ controls how much inertia the field has. The second term assigns energy to spatial differences. A field that changes sharply between nearby locations has a large gradient and therefore a large spatial energy. The parameter κ controls the strength of the coupling between nearby locations. The third term assigns energy to displacement from the baseline $\phi = 0$. The parameter λ controls the strength of the restoring force that pulls the field back toward that baseline. The signs follow the usual mechanical form $T - V$: the kinetic contribution is positive, while the spatial and restoring potential energies are subtracted.

A.4 The Action

The Lagrangian density describes one location at one moment. To describe the complete evolution of the field, it is integrated over the full spatial domain and time interval. This integral is called the *action*:

$$\mathcal{S}[\phi] = \int_{t_0}^{t_1} \int_{\Omega} \mathcal{L}\left(\partial_t \phi, \nabla \phi, \phi\right)\, d\mathbf{x}\, dt. \tag{18}$$

The inner integral combines the local contributions from every location in Ω. The outer integral combines those contributions over time. The notation $\mathcal{S}[\phi]$ emphasizes that the action depends on the entire history of the field, not only on its value at one time.

The principle of stationary action states that the physically realized field trajectory makes the action stationary with respect to small changes in the field. Applying this principle produces the field Euler–Lagrange equation:

$$\frac{\partial \mathcal{L}}{\partial \phi} - \partial_t \left(\frac{\partial \mathcal{L}}{\partial(\partial_t \phi)}\right) - \nabla \cdot \left(\frac{\partial \mathcal{L}}{\partial(\nabla \phi)}\right) = 0. \tag{19}$$

This is the field-theoretic analogue of the Euler–Lagrange equation used in ordinary mechanics and Lagrangian neural networks (Cranmer et al., 2020).

A.5 Deriving the Conservative Field Equation

We can apply Equation 19 to the Lagrangian density in Equation 17. Each term can be calculated separately.

The derivative with respect to the field itself is

$$\frac{\partial \mathcal{L}}{\partial \phi} = -\lambda \phi.$$

The derivative with respect to the temporal derivative is

$$\frac{\partial \mathcal{L}}{\partial(\partial_t \phi)} = \rho\, \partial_t \phi.$$

Taking one more temporal derivative gives

$$\partial_t \left(\frac{\partial \mathcal{L}}{\partial(\partial_t \phi)} \right) = \rho \, \partial_{tt} \phi.$$

The derivative with respect to the spatial gradient is

$$\frac{\partial \mathcal{L}}{\partial(\nabla \phi)} = -\kappa \nabla \phi.$$

Taking its divergence gives

$$\nabla \cdot \left(\frac{\partial \mathcal{L}}{\partial(\nabla \phi)} \right) = -\kappa \nabla \cdot (\nabla \phi) = -\kappa \nabla^2 \phi.$$

Substituting these expressions into the field Euler–Lagrange equation gives

$$-\lambda \phi - \rho \, \partial_{tt} \phi + \kappa \nabla^2 \phi = 0.$$

Multiplying both sides by (-1) produces the more familiar form

$$\underbrace{\rho \, \partial_{tt} \phi}_{\text{inertia}} - \underbrace{\kappa \nabla^2 \phi}_{\text{spatial coupling}} + \underbrace{\lambda \phi}_{\text{restoring force}} = 0. \tag{20}$$

Equation 20 describes a conservative field: energy can move between kinetic, spatial, and restoring components, but it is not removed from or added to the system. This form is a standard starting point in classical field theory and underlies familiar systems such as vibrating strings, elastic media, and wave equations, where dynamics follow from the exchange of energy among local kinetic and potential terms (Nastase, 2019; Strauss, 2008).

A.6 Adding Damping and External Forcing

Real systems are rarely perfectly conservative. Friction, viscosity, electrical resistance, and other dissipative mechanisms remove energy from a system. A common model of dissipation adds a term proportional to the field velocity:

$$\gamma \, \partial_t \phi.$$

The coefficient $\gamma \geq 0$ controls how quickly motion is damped. Larger values of γ cause disturbances to disappear more rapidly.

A system may also receive input from outside its current state. This input is represented by the forcing function

$$f(\mathbf{x}, t).$$

Forcing may be localized in space and time. A struck drum, a heated point on a metal plate, or a drop falling onto water each introduces energy at a particular location and moment.

Adding damping and forcing to Equation 20 gives

$$\underbrace{\rho \, \partial_{tt} \phi}_{\text{inertia}} + \underbrace{\gamma \, \partial_t \phi}_{\text{damping}} - \underbrace{\kappa \, \nabla^2 \phi}_{\text{spatial coupling}} + \underbrace{\lambda \phi}_{\text{restoring force}} = \underbrace{f(\mathbf{x}, t)}_{\text{external forcing}} . \tag{21}$$

Equation 21 is not a universal physical law. However, it is a useful template containing several common mechanisms:

- **Inertia** allows changes in the field to persist.
- **Damping** weakens ongoing motion.
- **Coupling** transfers disturbances between nearby locations.
- **Restoration** pulls the field toward a baseline.
- **Forcing** introduces new activity into the system.

Different choices of the coefficients produce different behaviors. Strong coupling spreads disturbances rapidly. Strong damping makes them disappear quickly. Strong restoration pulls the field back toward its equilibrium. Inertia allows the field to overshoot, oscillate, or carry motion forward.

A.7 Example: A Raindrop Striking Water

Consider a raindrop falling onto an otherwise still water surface. Let $\phi(\mathbf{x}, t)$ represent the displacement of the surface from its resting height.

At the moment of impact, the drop creates a localized forcing $f(\mathbf{x}, t)$. The water near the impact point is displaced downward and then upward. Because nearby parts of the water surface are coupled, this disturbance spreads outward.

The coupling term $-\kappa\nabla^2\phi$ transfers the disturbance across nearby locations. The inertial term $\rho\,\partial_{tt}\phi$ allows the surface to continue moving after the initial impact. This is why the water can overshoot its resting height and form an oscillating wave. The damping term $\gamma\,\partial_t\phi$ gradually removes energy, causing the expanding waves to become smaller. The restoring term $\lambda\phi$ pulls the surface back toward its equilibrium height.

The resulting wave is not programmed separately at every point on the water. Instead, it emerges because the same local dynamical equation is applied throughout the field.

A.8 From Continuous Fields to Field Neural Networks

Field Neural Networks transfer this idea from a continuous physical domain to a finite collection of interacting entities. The continuous field $\phi(\mathbf{x}, t)$ becomes a latent node-wise field $H_t \in \mathbb{R}^{N\times q}$.

Each of the N entities takes the role of one location in the domain, and each row of H_t stores its q-dimensional latent field state.

The continuous Laplacian $\nabla^2\phi$ is replaced by a graph operator applied to the latent field:

$$\nabla^2\phi \quad \longrightarrow \quad -L_tH_t.$$

The graph operator compares the state of each entity with the states of entities connected to it. The continuous forcing function becomes the event-derived forcing field:

$$f(\mathbf{x}, t) \quad \longrightarrow \quad F_t.$$

Finally, continuous time is divided into event intervals, and the differential equation is converted into a discrete update:

$$\phi(\mathbf{x}, t) \quad \longrightarrow \quad H_t.$$

The FNN update therefore retains the mechanisms of classical field dynamics while applying them to an unobserved state distributed across interacting entities. Unlike a physical field solver, the model is not supervised using measurements of ϕ or its derivatives. Its latent field and dynamical parameters are learned through their usefulness for the downstream prediction task.

B Derivation of the First-Order FNN

The general relational field equation introduced in Equation 3 is

$$\rho\,\partial_{tt}H(t) + c\,\partial_tH(t) + \kappa\,L_tH(t) + \lambda H(t) = F_t. \tag{3}$$

Here, $H(t) \in \mathbb{R}^{N\times q}$ is the latent field distributed over the N entities. The graph operator L_t describes relational coupling, F_t represents event forcing, and the coefficients control the relative strength of the field mechanisms. We use c for the coefficient of the first temporal derivative in this derivation so that it is not confused with the state-decay parameter used in the implemented first-order model later.

Removing the inertial term. The coefficient ρ multiplies the second temporal derivative $\partial_{tt}H(t)$. This term allows the field to maintain an independent velocity or momentum. A first-order model assumes that no such independent velocity state is required. The rate of change of the field is determined directly by its current state, relational coupling, and current forcing.

Setting $\rho = 0$ in Equation 3 gives

$$c\,\partial_tH(t) + \kappa L_tH(t) + \lambda H(t) = F_t \tag{22}$$

Solving for the field derivative. To obtain an evolution equation, we isolate the temporal derivative:

$$c\,\partial_t H(t) = F_t - \kappa L_t H(t) - \lambda H(t). \tag{23}$$

Assuming $c > 0$, division by c gives

$$\partial_t H(t) = -\frac{\lambda}{c} H(t) - \frac{\kappa}{c} L_t H(t) + \frac{1}{c} F_t. \tag{24}$$

Only the ratios between these coefficients determine the instantaneous evolution. We may therefore absorb the common scale $1/c$ into the remaining terms. Define

$$\delta_H = \frac{\lambda}{c}, \qquad \kappa_H = \frac{\kappa}{c}, \qquad \widetilde{F}_t = \frac{1}{c} F_t. \tag{25}$$

The normalized first-order relational dynamics are then

$$\partial_t H(t) = -\delta_H H(t) - \kappa_H L_t H(t) + \widetilde{F}_t. \tag{26}$$

For readability, the main text drops the subscripts and tildes and writes these normalized quantities as κ, δ, and F_t. Thus,

$$\partial_t H(t) = -\delta H(t) - \kappa L_t H(t) + F_t. \tag{27}$$

The parameter δ in Equation 27 is a state-decay coefficient. It should not be confused with γ velocity-proportional damping in a second-order field equation. In the first-order setting there is no independent velocity field to dampen.

Discrete-time approximation. Interaction streams are processed over discrete intervals rather than in continuous time. Let Δt denote the duration, or effective integration step, between successive field states. The temporal derivative can be approximated using a forward finite difference:

$$\partial_t H(t) \approx \frac{H_{t+1} - H_t}{\Delta t}. \tag{28}$$

Substituting Equation 28 into Equation 27 gives

$$\frac{H_{t+1} - H_t}{\Delta t} = -\delta H_t - \kappa L_t H_t + F_t. \tag{29}$$

Multiplying by Δt and solving for H_{t+1} yields

$$H_{t+1} = H_t + \Delta t \left(-\delta H_t - \kappa L_t H_t + F_t\right). \tag{30}$$

Equation 30 is the first-order Field Neural Network update and reproduced as Equation 9 in the main paper. It is an explicit Euler discretization of the normalized relational field equation.

Interpretation of the update. The update contains three mechanisms:

- **State decay:** The term $-\delta H_t$ gradually returns the latent field toward zero in the absence of coupling and forcing.
- **Relational diffusion:** The term $-\kappa L_t H_t$ reduces degree-normalized relational disagreement across connected entities. Larger values of κ cause latent activity to propagate more strongly across the event-induced graph.
- **Event forcing:** The term F_t introduces new activity derived from the events observed during the current interval.

C Derivation of the Second-Order FNN

Again, the general relational field equation introduced in Equation 3 is

$$\rho\,\partial_{tt} H(t) + \gamma\,\partial_t H(t) + \kappa L_t H(t) + \lambda H(t) = F_t. \tag{3}$$

Unlike the first-order model, the second-order model retains the second temporal derivative $\partial_{tt} H(t)$. The field may therefore possess an independent rate of change that continues to affect later states even after the current forcing has been removed.

Introducing a velocity-like field. Define the velocity-like latent field

$$V(t) = \partial_t H(t). \tag{31}$$

Because $V(t)$ is the first temporal derivative of $H(t)$, its derivative is the second derivative of the field:

$$\partial_t V(t) = \partial_{tt} H(t). \tag{32}$$

Substituting these definitions into Equation 3 gives

$$\rho\, \partial_t V(t) + \gamma V(t) + \kappa L_t H(t) + \lambda H(t) = F_t, \tag{33}$$

$$\partial_t H(t) = V(t). \tag{34}$$

The second-order equation has therefore been rewritten as two coupled first-order equations: one evolves the velocity-like field $V(t)$, and the other uses that velocity to evolve $H(t)$.

Normalizing the inertial coefficient. Solving Equation 33 for the velocity derivative gives

$$\partial_t V(t) = -\frac{\gamma}{\rho} V(t) - \frac{\kappa}{\rho} L_t H(t) - \frac{\lambda}{\rho} H(t) + \frac{1}{\rho} F_t. \tag{35}$$

The common factor $1/\rho$ can be absorbed into the remaining coefficients. Define

$$\bar{\gamma} = \frac{\gamma}{\rho}, \qquad \bar{\kappa} = \frac{\kappa}{\rho}, \qquad \bar{\lambda} = \frac{\lambda}{\rho}, \qquad \bar{F}_t = \frac{1}{\rho} F_t. \tag{36}$$

The normalized continuous-time system is then

$$\partial_t V(t) = -\bar{\gamma} V(t) - \bar{\kappa} L_t H(t) - \bar{\lambda} H(t) + \bar{F}_t, \tag{37}$$

$$\partial_t H(t) = V(t). \tag{38}$$

For readability, we drop the bars below and write the normalized quantities as γ, κ, λ, and F_t.

Local restoration. The physical second-order FNN retains the local restoring term. Substituting $\omega^2 = \lambda$ is a more familiar form:

$$\partial_t V(t) = -\gamma V(t) - \kappa L_t H(t) - \omega^2 H(t) + F_t, \tag{39}$$

$$\partial_t H(t) = V(t). \tag{40}$$

The graph-coupling term complements this local restoration by reducing relational disagreement: when an entity's latent state differs from the states to which it is coupled, $L_t H(t)$ changes its velocity toward those surroundings.

Canonical velocity discretization. Using the forward finite-difference approximation

$$\partial_t V(t) \approx \frac{V_{t+1} - V_t}{\Delta t}, \tag{41}$$

and substituting it into Equation 39 gives

$$\frac{V_{t+1} - V_t}{\Delta t} = -\gamma V_t - \kappa L_t H_t - \omega^2 H_t + F_t. \tag{42}$$

Multiplying by Δt and solving for V_{t+1} yields the canonical Euler velocity update

$$V_{t+1} = V_t - \Delta t \left(\gamma V_t + \kappa L_t H_t + \omega^2 H_t\right) + \Delta t F_t. \tag{43}$$

The unit coefficient on V_t in Equation 43 follows directly from the Euler discretization.

Physical velocity recurrence. Equation 43 is the second-order update used in the physical FNN: previous velocity has the Euler coefficient one, and damping enters only through $(1-\gamma\Delta t)V_t$; $\omega^2 H_t$ supplies local harmonic restoration. We do not use an independent learned velocity-persistence coefficient in the physical experiments.

Updating the field. The field derivative satisfies $\partial_t H(t) = V(t)$. A fully explicit Euler update would use the previous velocity:

$$H_{t+1} = H_t + \Delta t V_t. \tag{44}$$

The second-order FNN instead uses the newly computed velocity:

$$H_{t+1} = H_t + \Delta t V_{t+1}. \tag{45}$$

The ordering of the position update is therefore semi-implicit, or symplectic-Euler-style: the velocity is first updated using the current field and velocity, and the new velocity is then used to advance the field.

Interpretation of the update. The implemented second-order FNN contains four distinguishable dynamical mechanisms:

- **Velocity persistence and damping:** The term $(1 - \gamma\Delta t)V_t$ carries velocity forward while damping it.
- **Relational coupling:** The term $-\Delta t\, \kappa L_t H_t$ changes the velocity according to relational disagreement in the current field.
- **Local restoration:** The term $-\Delta t\, \omega^2 H_t$ pulls the field toward its baseline.
- **Event forcing:** The term $\Delta t F_t$ changes the velocity in response to events in the current interval.

The explicit velocity field supplies second-order memory. Event forcing or relational coupling can alter V_t, and the resulting change may continue to influence later field states after the original event has passed.

Velocity initialization. Each physical episode starts at zero field and zero velocity. Subsequent velocities are stored and updated by Equation 43.

D Experimental Methodology

This section specifies the synthetic generator, event-only prediction task, and controlled ablations. The details below define the quantities needed to reproduce and interpret the reported experiments.

D.1 Hidden force-field simulator

Each episode is an independent simulation trajectory on the same fixed, unobserved topology. It evolves a hidden field $\mathbf{h}_t[i] \in \mathbb{R}^d$, with $d = 4$ and $|V| = N$. Second-order systems additionally have velocity $\mathbf{v}_t[i] \in \mathbb{R}^d$. For each directed support edge $i \to j$, the simulator produces the physical pair force

$$\mathbf{f}_{i\to j,t} = sA_{ij}\big(\mathbf{h}_t[i] - \mathbf{h}_t[j]\big), \tag{46}$$

where A is the hidden binary support and s is the force scale. The total force applied to node j is

$$\mathbf{q}_t[j] = \sum_i \mathbf{f}_{i\to j,t} + \mathbf{r}_t[j], \tag{47}$$

where $\mathbf{r}_t$ is an external impulse. Each episode starts from $\mathbf{h}_0 = \mathbf{v}_0 = 0$; an impulse is applied at its first bin and again every 12 bins. Its node is selected uniformly among active nodes and its magnitude is sampled uniformly from $[0.9, 1.3]$ in a fixed field direction.

Writing $L_A = D - A$ for the unnormalized Laplacian of the symmetric hidden graph, where $D_{ii} = \sum_j A_{ij}$, the total incoming force is $\mathbf{q}_t = -sL_A\mathbf{h}_t + \mathbf{r}_t$. Diffusion is therefore the first-order field update in Eq. 30, with $\delta = \gamma$, $\kappa L_t = sL_A$, and $F_t = \mathbf{r}_t$. Wave and coupled-oscillator systems use the semi-implicit second-order updates in Eqs. 43 and 45, with the same coupling and forcing substitution. The restoring term $\omega^2\mathbf{h}_t$ is omitted for first-order diffusion. In the main simulation experiments, these coefficients are fixed during model training.

Table A1: Default synthetic benchmark configuration. The coupled-oscillator task uses the same scripted benchmark values as wave in Experiment 1; the canonical generator values used in controlled recovery are recorded separately in the recovery manifest.

Quantity	Benchmark value
Nodes, field dimension	$N = 64$, $d = 4$
Episodes, bins per episode	10, 72
Train/validation/test episodes	60% / 20% / 20%
Integration step	$\Delta t = 0.10$
Damping, restoring frequency, force scale	$\gamma = 0.15$, $\omega = 0.80$ (second order), $s = 0.80$
Impulse interval	12 bins
Training epochs, seeds	20, five (0–4)
Ranking candidates	one positive plus 10 sampled destinations

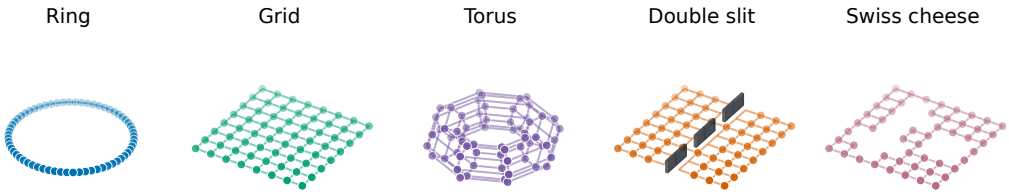

Figure A1: Graph topologies used in the synthetic experiments: ring, grid, torus, double slit, and Swiss cheese. Nodes represent field locations and edges indicate pairwise coupling. The double-slit barrier restricts connections to two openings, while Swiss cheese removes interior regions. The 3D layouts illustrate connectivity, not simulated field values.

D.2 Topologies and observed event stream

We use ring, grid, torus, double-slit, and Swiss-cheese supports (Fig. A1). The ring has two periodic nearest neighbors per node. The remaining supports are derived from an 8×8 nearest-neighbor grid; the torus makes both axes periodic, the double-slit inserts a barrier with apertures, and Swiss cheese removes interior regions. These choices change A, but not Eqs. 46–47.

The model receives only directed event batches $(i, j, \mathbf{f}_{i \to j,t})$. It never receives A, $\mathbf{h}_t$, $\mathbf{v}_t$, or the generator coefficients. An internal pair is emitted when $\|\mathbf{f}_{i \to j,t}\|_2 > \tau$. If more pairs qualify than the measurement budget, the simulator retains the largest-magnitude forces. Each external impulse is emitted as a self-event and marked as external. Consequently, thresholding changes observation sparsity only; it does not change the hidden physical evolution.

The 60/20/20 split is by whole independent episodes, not by individual events or time bins. This prevents an episode trajectory from appearing in more than one split while preserving temporal order within every trajectory.

D.3 Baselines, objectives, and evaluation

The main synthetic panel contains the FNN and six baselines: GRU, DeepSets, SetTransformer, modern Hopfield, LNN, and HNN. All methods receive the same event stream and are trained to rank the next active interaction and, where the model exposes a force decoder, to predict its next force vector.

For each positive next-bin event, training samples ten alternative destinations and minimizes a sampled softmax ranking loss. Physical event models add a force-regression loss. Force targets are normalized using the training-split channel standard deviation, with additional weight on large forces; this normalization is used only in the loss and is not an input to the model. We report force MSE, MRR and Hits@K, plus event AUROC/AUPRC from a separate balanced active-versus-inactive diagnostic. Closed-loop evaluation rolls predicted force events forward while retaining the future pair-query schedule; this measures dynamics propagation without claiming to generate the future event schedule itself.

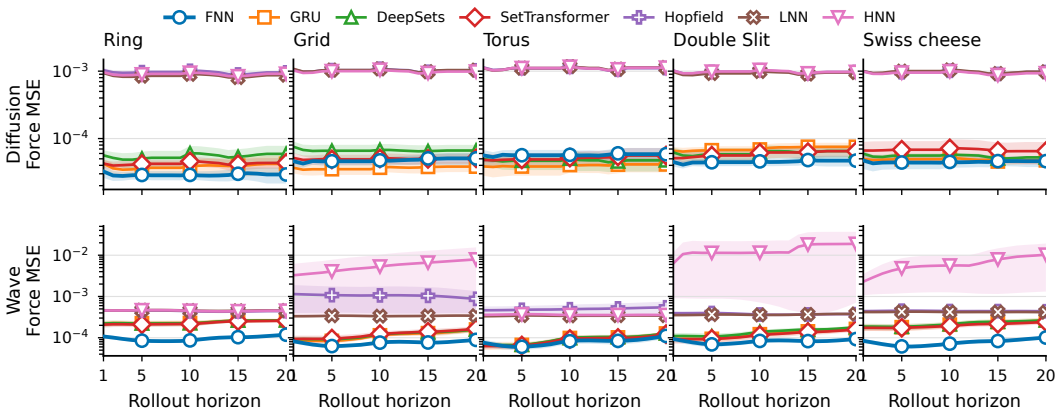

Figure A2: Topology-resolved force-prediction error over closed-loop rollout horizon. Rows show diffusion and wave dynamics, and columns show the five hidden topologies. Curves summarize five seeds; lower is better.

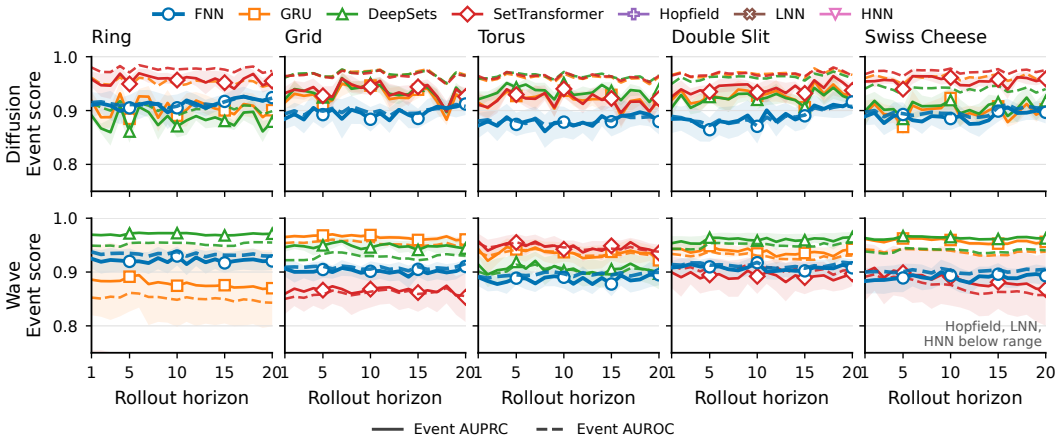

Figure A3: Topology-resolved event discrimination over closed-loop rollout horizon. Solid curves show AUPRC and dashed curves show AUROC. Rows show dynamics and columns show topologies; curves summarize five seeds.

E Synthetic Experiments

Dynamics and topology benchmark. The benchmark crosses diffusion and wave dynamics with the five topologies, seven models, and five seeds. It uses the fixed configuration in Table A1 and a 20-step rollout.

Figures A2 and A3 unpack the compact summary in Figure 3. Rather than pooling over topology, each column retains one hidden geometry; rows distinguish diffusion from wave dynamics. The figures therefore show whether a model's aggregate rollout performance is consistent across propagation environments.

Rollout-training length. One-step training does not directly penalize errors that accumulate when predictions are fed back into the model. We therefore test whether training through longer rollouts improves long-horizon prediction. We vary the number of differentiable training steps, $K_{\text{train}} \in \{1, 2, 5, 10, 20\}$, while evaluating every model at horizon 20, using diffusion and wave dynamics across the five topologies. Holding the evaluation horizon fixed allows us to compare training rollout lengths on the same prediction task. Figures A4 and A5 report force-prediction error and event-discrimination performance, respectively.

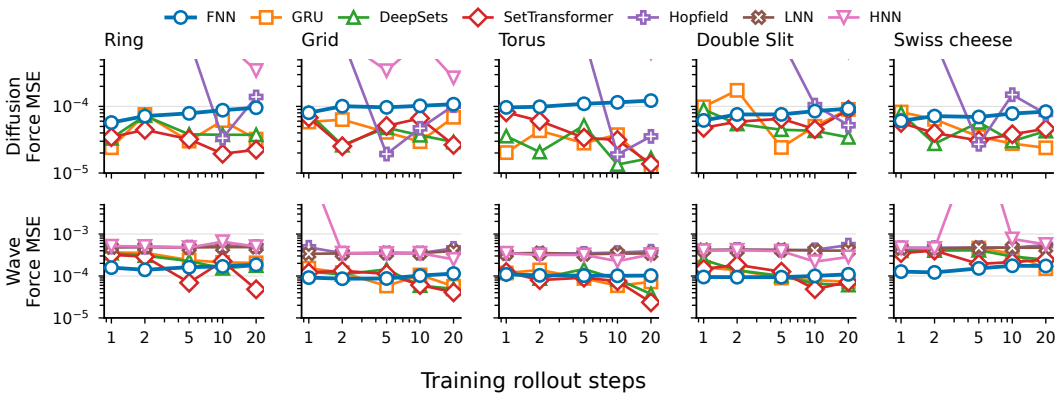

Figure A4: Effect of training rollout length on force-prediction error at evaluation horizon 20. Rows show diffusion and wave dynamics, columns show topologies, and the horizontal axis is K_{train}. Lower force MSE is better.

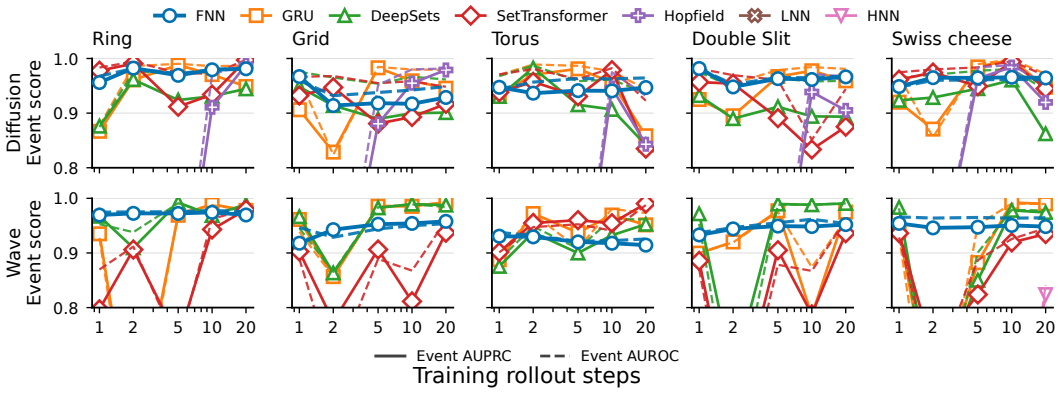

Figure A5: Effect of training rollout length on event discrimination at evaluation horizon 20. Solid curves show AUPRC and dashed curves show AUROC; rows and columns match Figure A4. Higher is better.

Observation threshold. Thresholding is calibrated separately for each dynamics/topology pair to produce dense, medium, sparse, and very-sparse regimes with comparable retained event fractions. We report both the literal threshold and observed events per bin. Training uses $K_{\text{train}} = 1$ and evaluation uses horizon 20. Figures A6 and A7 show the resulting robustness curves for various models. Each column fixes a hidden topology, each row fixes the dynamics, and the horizontal axis reports the calibrated retained-event fraction.

Changing the observation threshold changes both the amount of supervision and the event label. At sparse thresholds, positives are the strongest currently active forces; at dense thresholds, nearly every support edge is positive, including weak residual interactions. The balanced event diagnostic contrasts current positives with currently unobserved pairs. Consequently, its dense regime increasingly measures recovery of persistent interaction support rather than only transient strong-event prediction. The FNN score contains a learned topology, so its increasing AUROC/AUPRC in Figure A7 is evidence of this structural recovery.

E.1 Physical-parameter recovery

The preceding benchmark experiments hold the FNN physical coefficients—the integration scale Δt, damping γ, restoring frequency ω, and pair-force scale s—fixed at deliberately non-generating values. This isolates learning of the persistent relational operator: predictive gains cannot arise merely because adaptable physical coefficients compensate for topology errors. It does not, however, test whether the complete FNN can recover the dynamics that generated the synthetic data.

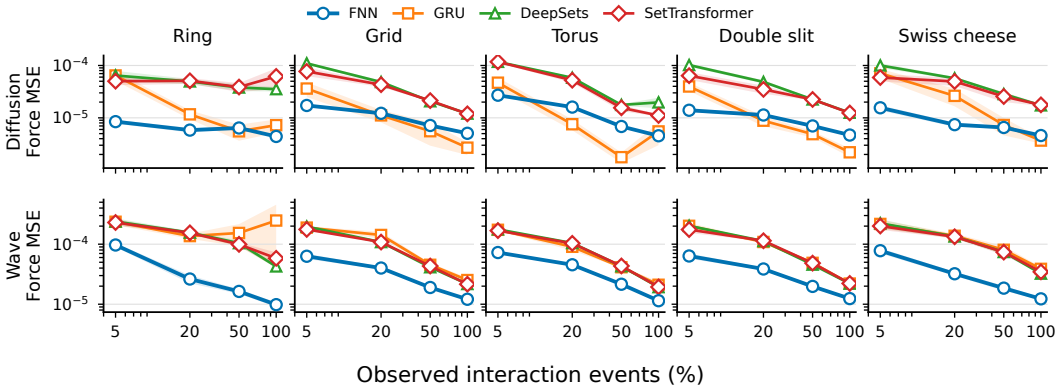

Figure A6: Force-prediction error under observation-threshold ablation. Rows show diffusion and wave dynamics, columns show hidden topologies, and the horizontal axis gives the retained interaction fraction. Curves show five-seed means with shaded ± 1 SEM; lower force MSE is better.

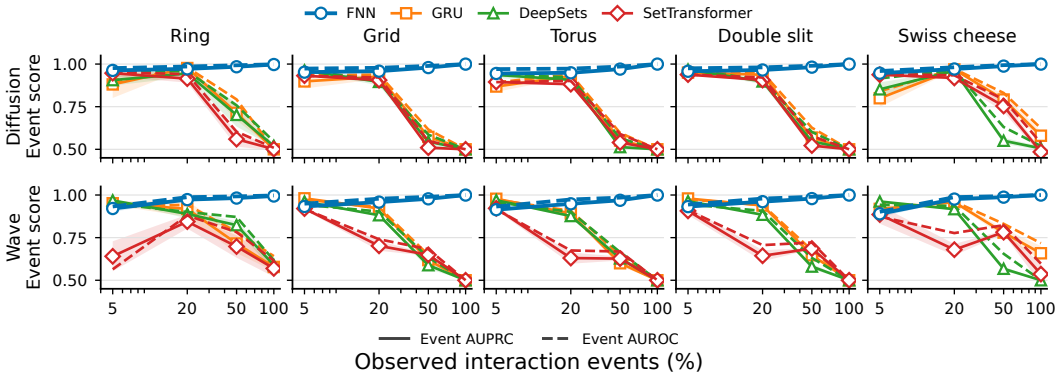

Figure A7: Event discrimination under observation-threshold ablation. Solid curves show AUPRC and dashed curves show AUROC; rows and columns match Figure A6. Curves show five-seed means with shaded ± 1 SEM; higher is better.

Oracle-topology identifiability test. We first fix the FNN graph operator to the known hidden support and initialize the physical coefficients over a grid of values spanning 0.25 to 4 times their generator values. This is intentionally an oracle setting. By removing uncertainty in the relational operator, it isolates identifiability of the physical coefficients under the same predictive objective used in the main experiments.

Figure A8 shows that joint optimization does not generally recover the generator coefficients. Instead, the trajectories approach off-target compensation manifolds: changing one coefficient can offset another while maintaining similar predictive behavior. Thus, the predictive objective identifies useful effective dynamics more reliably than it identifies every scalar coefficient under simultaneous updates.

Alternating oracle recovery. We next update one physical coefficient at a time while fixing the topology and all remaining coefficients. This prevents an immediately updated scalar from being counterbalanced by another scalar during the same optimization block. Figure A9 shows that this schedule moves dispersed initializations toward the generator ratios in the oracle-topology setting. It therefore establishes conditional recovery: the physical parameterization is recoverable when uncertainty in the relational operator is removed and coefficient compensation is controlled.

Non-oracle synthetic recovery. Finally, we remove the oracle topology and initialize both the persistent topology gates and physical coefficients away from their generator values. For each diffusion– or wave–topology–seed combination, we alternate a topology block with scalar-specific

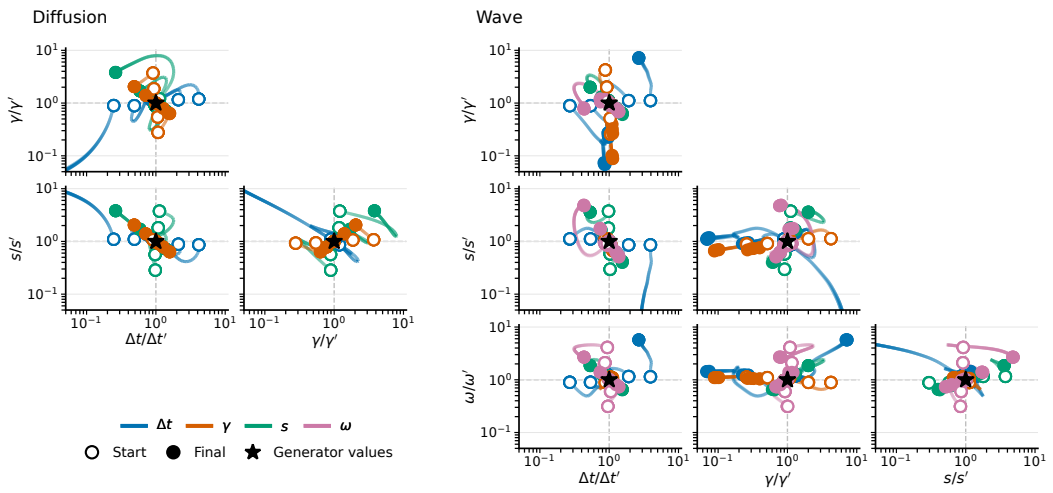

Figure A8: Joint physical-parameter recovery with oracle topology. Each panel compares two learned-to-true ratios for diffusion or wave dynamics. Open circles denote initializations, filled circles denote final values, and the star at $(1, 1)$ marks exact recovery. Joint trajectories converge to predictive but generally off-target configurations.

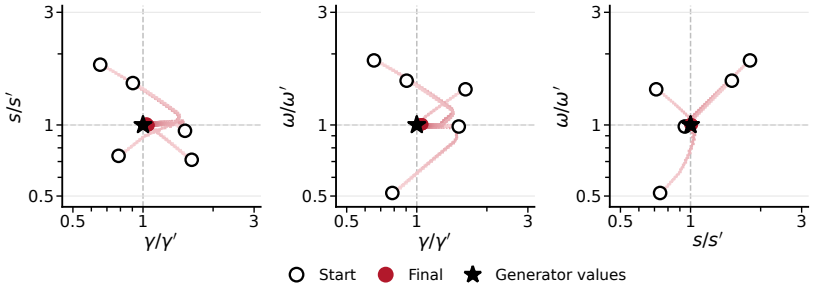

Figure A9: Alternating physical-parameter recovery for wave dynamics on a ring with fixed topology. Each panel plots two learned-to-generator value ratios. Open circles denote random initializations, filled circles denote final values, and stars mark the generator value $(1, 1)$. Paths darken as training progresses.

physical blocks, repeating the full cycle four times. Diffusion updates Δt, γ, and s; wave dynamics additionally update ω.

Figure A10 shows that this non-oracle procedure retains strong held-out event discrimination and hidden topology recovery across the five supports. Figure A11 reports the final physical coefficients relative to their known generator values. The learned coefficients generally occupy the correct scale, although recovery varies by coefficient and topology. This variation is expected when topology and dynamics must both be inferred: residual changes in topology gates and physical scalars can still partially compensate while preserving predictive performance.

F Real-World Experiments

This section details the evaluation protocol underlying Section 3.3 and examines which components of FNN contribute to its predictions. Unlike the synthetic experiments, these datasets provide neither a known physical field nor generating coefficients. The learned dynamics are therefore evaluated as a predictive representation, not as recovered physical laws.

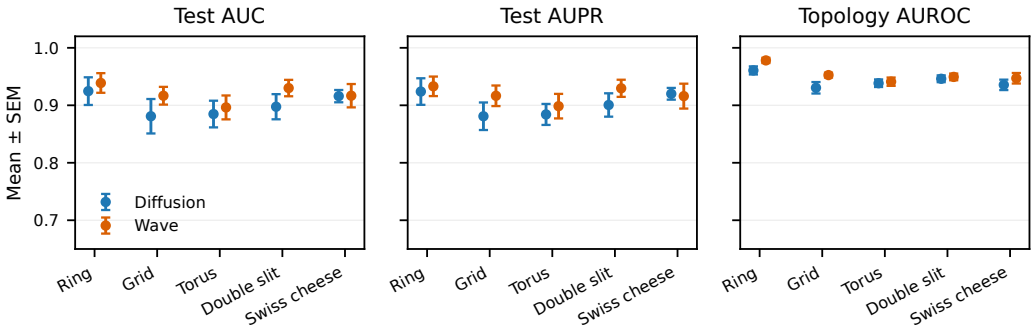

Figure A10: Predictive and structural performance under non-oracle alternating recovery. Points show seed means and error bars show ± 1 SEM. Diffusion and wave points are horizontally offset within each topology. Panels report held-out event AUROC, event AUPRC, and hidden-topology AUROC; higher is better.

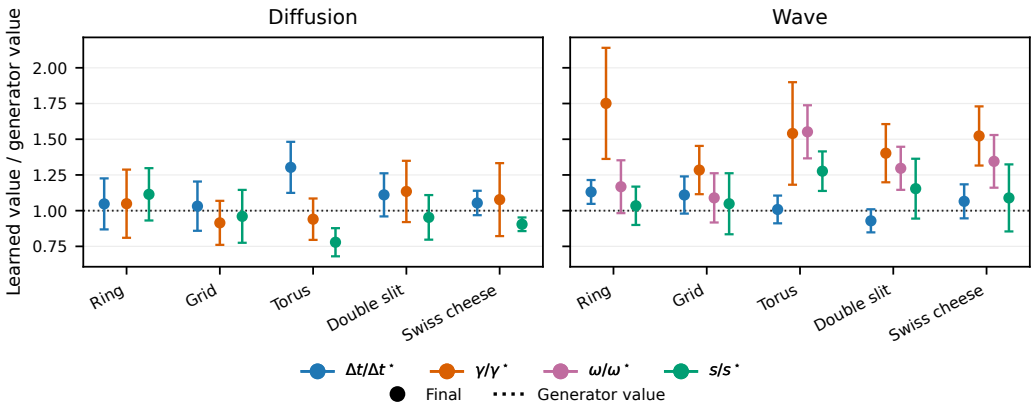

Figure A11: Final physical calibration after non-oracle alternating recovery. Each point is the final learned coefficient divided by its generator value, with ± 1 SEM across seeds. Colors distinguish $\Delta t/\Delta t^\star$, $\gamma/\gamma^\star$, $\omega/\omega^\star$, and $s/s^\star$; ω applies only to wave dynamics. The dotted reference denotes exact recovery.

F.1 Datasets and prediction task

We use the published DyGLib datasets Enron, Social Evolution, UCI, Flights, Canadian Parliament, US Legislature, UN Trade, UN Vote, and Contact (Yu et al., 2023). These datasets describe interactions within a single node population and span communication, proximity, transportation, political, and economic networks. We use the benchmark's processed data rather than substituting similarly named datasets from other sources.

The task is temporal link prediction: distinguish an observed interaction from a sampled negative using the preceding interaction history. We retain DyGLib's chronological splitting procedure, with timestamp boundaries at the 70th and 85th percentiles. Its new-node protocol additionally withholds selected nodes and their incident training interactions.

The main comparison uses random negative sampling, with one sampled negative per positive interaction. This is a binary link-prediction task, not the ten-alternative ranking task used in the synthetic experiments. Likewise, new-node evaluation and inductive negative sampling are distinct protocols: the former changes which positive interactions are evaluated, whereas the latter changes how negatives are constructed.

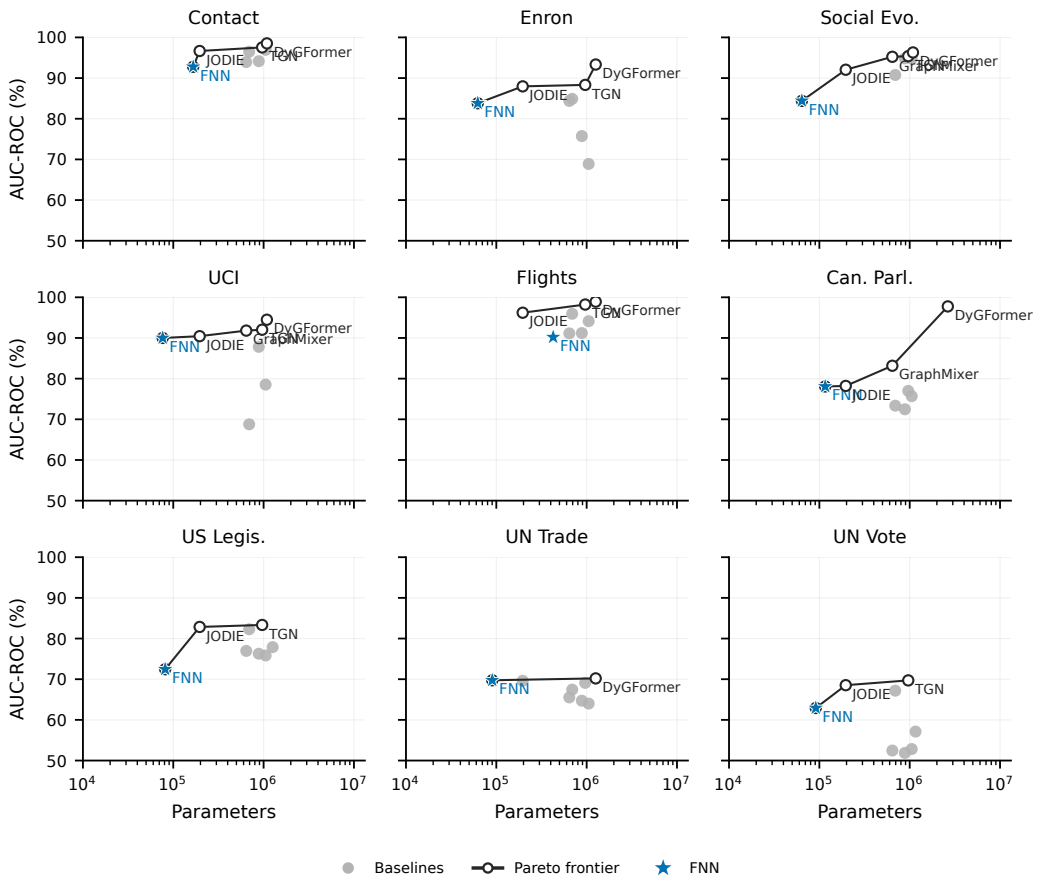

Figure A12: AUROC versus parameter count across all nine real-world datasets. Stars identify FNN; gray points show comparison models with available parameter counts. Horizontal axes are logarithmic. Connected open circles mark the accuracy–parameter frontier: no included alternative achieves at least as high an AUROC with no more parameters, with a strict improvement in either quantity.

F.2 FNN INTEGRATION INTO DYGLIB

FNN is integrated as a memory-based backbone in DyGLib's native training and evaluation pipeline. Each node maintains a q-dimensional field and velocity, (H, V), initialized to zero. An observed interaction supplies a unit input at its destination, weighted by a persistent sigmoid gate for the interacting pair. For multichannel fields, a learned shared drive vector distributes this input across channels. The adapter uses node identities, interaction directions, and timestamp ordering; it does not use node or edge attributes as predictive inputs—although it could.

Learnable pair gates are allocated from the training graph only. DyGLib's training neighbor representation supplies both directions of each observed connection; validation and test interactions do not introduce additional learned pair parameters. These gates weight observed inputs rather than directly supplying link scores.

The default backbone uses the local damped, second-order update from Eqs. 43 and 45. Damping and restoring frequency are learned per channel, together with a shared input scale. The integration step is fixed at $\Delta t = 0.1$ per distinct observed timestamp. Thus, the default model uses an event clock: timestamps determine ordering and simultaneity, but elapsed gaps do not determine the integration step. Additional propagation and elapsed-time variants are treated as separate ablations.

Table A2: Selected FNN hyperparameters for the real-world experiments. Here q is the number of field channels, and propagation denotes the number of input-propagation hops.

Dataset	q	Learning rate	Weight decay	Batch size	Propagation
Enron	8	0.003	0.0001	25	0
Social Evo.	4	0.001	0	50	0
UCI	8	0.003	0.0001	50	1
Flights	8	0.001	0.001	200	2
Can. Parl.	4	0.001	0.001	50	0
US Legis.	4	0.001	0	50	0
UN Trade	4	0.001	0.001	50	0
UN Vote	4	0.001	0.001	50	0
Contact	1	0.001	0	50	1

For prediction, a learned linear projection maps the concatenated node state $[H[i], V[i]]$ into DyGLib's embedding space. Its standard two-layer link decoder then scores the source and destination embeddings. There is no auxiliary force-regression objective in these experiments.

F.3 Training, evaluation, and reporting

Training uses DyGLib's binary cross-entropy objective and jointly optimizes the field coefficients, pair gates, input mapping, and prediction head. Memory is carried forward chronologically and detached between training batches.

Checkpoint selection follows the native validation rule: a checkpoint is replaced when both validation AP and AUROC are no worse than their previously accepted values. Final test performance is obtained from the selected checkpoint, not the final training epoch.

We retain DyGLib's metric aggregation, which computes AP and AUROC within evaluation batches and averages the resulting batch scores. Our repeated-run summaries report means and SEM across seeds. These uncertainties describe variation between training runs, not uncertainty over independently sampled datasets.

The comparison-model scores in Figure 4 are taken from the published DyGLib benchmark; they are not newly rerun baselines. FNN is evaluated using the corresponding data and random-negative protocol. Published baseline uncertainties retain their source convention and should not be interpreted automatically as SEM.

Figure A12 extends the three-dataset comparison in Figure 5 to all nine real-world datasets. This broader view shows where FNN's compact parameterization offers a competitive trade-off and where larger models achieve substantially higher AUROC. The frontiers are relative to the included models with available parameter counts.

Hyperparameters were selected separately for each dataset using Asynchronous Successive Halving Li et al. (2020). The search varied the field dimension q, learning rate, weight decay, batch size, and number of input-propagation hops. Table A2 lists the selected configurations used for the repeated-seed experiments. Propagation zero disables spreading of event inputs; positive values specify the number of hops over the training graph. This propagates incoming activity, rather than applying a Laplacian directly to the existing latent field.

F.4 Which components support prediction?

Learned connection weights and dynamical coefficients. Table A3 compares the full model with two retrained ablations. The first fixes every represented pair gate to 0.5; the second fixes damping, restoring frequency, and input scale at initialization. Uniform gates produce small changes across the evaluated datasets, suggesting that learning individual connection weights contributes little under this configuration. This does not remove the interaction history or the graph on which inputs are defined.

Table A3: Ablations of learned topology weighting and dynamical parameters for transductive dynamic link prediction. Values are test AP (%) reported as mean ± SEM. The uniform-topology ablation fixes every candidate-edge gate to 0.5; the dynamical ablation fixes γ, ω, and the input scale at their initial values. Parentheses report the change from the full FNN in percentage points, so negative values indicate degradation.

Dataset	Full FNN	Uniform topology gate	Fixed γ, ω, input scale
Enron	82.55 ± 0.08	82.45 ± 0.17 (-0.10)	81.01 ± 0.01 (-1.54)
Social Evo.	80.45 ± 0.04	80.51 ± 0.10 $(+0.06)$	79.55 ± 0.06 (-0.90)
UCI	91.01 ± 0.09	91.03 ± 0.12 $(+0.02)$	87.11 ± 0.19 (-3.90)
Flights	90.03 ± 0.03	89.98 ± 0.06 (-0.05)	88.70 ± 0.69 (-1.33)
Can. Parl.	79.48 ± 0.18	79.14 ± 0.40 (-0.34)	79.53 ± 0.12 $(+0.05)$
US Legis.	68.78 ± 0.16	69.07 ± 0.12 $(+0.29)$	69.11 ± 0.24 $(+0.33)$
UN Trade	65.84 ± 0.03	65.78 ± 0.11 (-0.06)	65.74 ± 0.06 (-0.10)
UN Vote	59.98 ± 0.03	60.00 ± 0.08 $(+0.02)$	59.95 ± 0.04 (-0.03)
Contact	90.36 ± 0.05	90.38 ± 0.04 $(+0.02)$	83.39 ± 0.10 (-6.97)

Table A4: Parameter-specific ablations of the learned dynamical coefficients on Contact and UCI. Values are test mean ± SEM (%). Fixing a parameter holds it at its initialization while the remaining parameters are learned. Contact is largely insensitive to fixing ω or the input scale, but performance drops sharply when γ is fixed. UCI is also most sensitive to γ, with a smaller but clear degradation when ω is fixed; fixing the input scale has negligible effect.

Configuration	Contact AP	Contact AUC	UCI AP	UCI AUC
Full FNN	90.36 ± 0.07	92.77 ± 0.03	91.12 ± 0.08	90.15 ± 0.17
Fix input scale	90.43 ± 0.04	92.84 ± 0.01	91.03 ± 0.06	90.09 ± 0.03
Fix ω	90.42 ± 0.07	92.81 ± 0.03	89.22 ± 0.58	87.45 ± 0.62
Fix γ	83.13 ± 0.07	85.68 ± 0.11	87.50 ± 0.17	84.96 ± 0.31
Fix γ, ω	83.31 ± 0.11	85.77 ± 0.15	87.10 ± 0.40	84.28 ± 0.56

Post-hoc sensitivity to restoring frequency. Table A5 perturbs ω only at evaluation time, leaving all other learned parameters fixed. Can. Parl. is nearly insensitive to either perturbation. UN Trade responds asymmetrically, with lower AP at 0.5ω and higher AP at 2ω, whereas UN Vote shows the opposite pattern and is especially sensitive to doubling ω. These interventions complement the retrained ablations above: the functional role of restoring frequency depends strongly on the interaction stream, and the learned numerical value should not be interpreted as a universally meaningful physical constant.

Direct access to velocity. Next, Table A6 intervenes on trained checkpoints. We zero the projection weights multiplying V while preserving the internal velocity state and its role in the field transition. This tests the trained predictor's reliance on direct velocity features, rather than the performance of a model retrained without them.

The resulting sensitivity differs from sensitivity to coefficient learning. UCI changes little when direct velocity access is removed, despite benefiting from learned dynamics. Conversely, UN Trade changes little when coefficients are fixed during training but loses 24.94 AP points when velocity is hidden from its trained readout. Figure A13 summarizes this distinction. A useful velocity state and a need to learn its governing coefficients are not the same property.

First- versus second-order dynamics. Table A7 compares separately trained first- and second-order models. Their predictive scores are often close, and neither order dominates every dataset. This is compatible with the checkpoint interventions: a retrained model can reorganize its representation, whereas removing a feature from an already trained readout offers no opportunity to compensate. The Canadian Parliament comparison controls the weight-decay setting because the original regularization affected the two orders differently.

Table A5: Post-hoc sensitivity to the learned restoring frequency. Each trained checkpoint is evaluated with ω replaced by 0.5ω or 2ω, without retraining. Values are five-seed mean test AP (%). Parentheses report the change from evaluation with the learned ω in percentage points.

Dataset	0.5ω	**Learned** ω	2ω
Can. Parl.	82.503 (+0.005)	82.498	82.473 (−0.025)
UN Trade	63.683 (−0.319)	64.002	65.040 (+1.038)
UN Vote	52.006 (+0.578)	51.428	48.559 (−2.869)

Table A6: Post-hoc reliance on direct velocity features. For each trained second-order FNN checkpoint, the velocity columns of the readout are set to zero while the second-order dynamics and internal velocity state remain unchanged. Values are five-seed test means (%). The reported drop is full-model score minus velocity-hidden score; this diagnostic measures reliance of the trained predictor on direct access to v, not the performance of a retrained position-only model.

Dataset	**Full AP**	v **hidden AP**	**AP drop**	**Full AUC**	v **hidden AUC**	**AUC drop**
Enron	82.55	82.34	0.21	83.60	83.42	0.18
Social Evo.	80.45	79.41	1.04	84.40	83.55	0.85
UCI	91.01	90.95	0.06	90.00	89.90	0.10
Flights	90.03	89.77	0.26	90.21	89.88	0.33
Can. Parl.	79.48	67.93	11.55	78.07	62.42	15.65
US Legis.	68.78	60.70	8.08	72.45	59.70	12.75
UN Trade	65.84	40.90	24.94	69.74	34.44	35.30
UN Vote	59.98	54.53	5.45	62.92	52.24	10.68
Contact	90.36	78.72	11.64	92.77	81.83	10.94

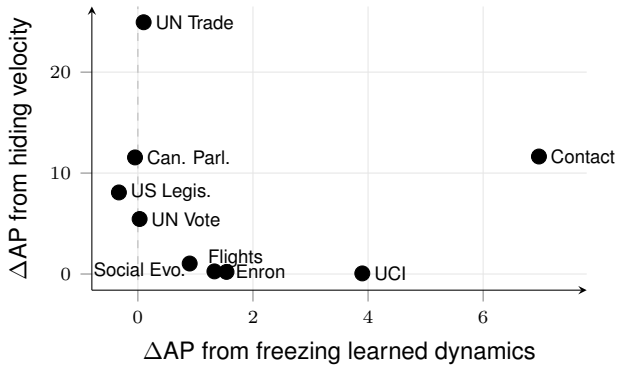

Figure A13: Mechanism map of FNN reliance across datasets. The horizontal axis shows the change in test AP when the learned dynamical parameters (γ, ω, and input scale) are frozen, and the vertical axis shows the AP loss when direct velocity access is hidden from the trained second-order readout. Positive x means freezing dynamics hurts; negative x means the ablation slightly improves AP. Points higher on the plot rely more on direct velocity features.

Table A7: First- versus second-order FNN test AP (%). Positive differences favor second order.

Dataset	**First order**	**Second order**	**2nd − 1st**
Enron	82.11 ± 0.05	82.55 ± 0.08	+0.44
Social Evo.	79.69 ± 0.16	80.45 ± 0.04	+0.76
UCI	90.96 ± 0.04	91.01 ± 0.09	+0.05
Flights	89.99 ± 0.01	90.03 ± 0.03	+0.05
Can. Parl.	78.19 ± 0.37	79.76	+1.57
US Legis.	69.84 ± 0.22	68.78 ± 0.16	−1.06
UN Trade	65.63 ± 0.08	65.84 ± 0.03	+0.21
UN Vote	58.97 ± 0.35	59.98 ± 0.03	+1.01
Contact	89.83 ± 0.04	90.36 ± 0.05	+0.53