1 Dataset Construction Details

1.1 Synthetic Meta-Dataset

To construct a synthetic meta-dataset that reflects the dynamics of both spring systems and charged particle systems, we begin by formalizing the governing equations of these two types of physical systems considered in NRI.

The spring system in NRI follows Hooke's law, with all springs having identical force constants σ, and the mass of each object is set to 1. The force between particles is proportional to the negative gradient of potential energy:

$$\dot{\mathbf{x}}_i = -\sigma \cdot a_{ij} \sum_{i \neq j} (\mathbf{x}_i - \mathbf{x}_j), \tag{1}$$

Here, $\mathbf{x}_i$ represents the state vector of the i-th object at a certain moment, a_{ij} denotes the element of the adjacency matrix A, where $a_{ij} = 1$ indicates that there is a spring connecting object i to object j, and $a_{ij} = 0$ indicates no such connection exists. The interaction graph is undirected, meaning that $a_{ij} = a_{ji}$ for all i and j.

Charged particle interactions follow Coulomb's law, in the charged particle system considered by NRI, all particles have the same magnitude of charge but differ only in sign, and the mass of each particle is set to 1.

$$\dot{\mathbf{x}}_i = \sigma \cdot \sum_{i \neq j} \frac{q_i q_j}{\|\mathbf{x}_i - \mathbf{x}_j\|^3} (\mathbf{x}_i - \mathbf{x}_j), \tag{2}$$

where $q_i, q_j \in \{1, -1\}$ are the electric charges of particles i and j, σ is the Coulomb constant, and the direction of the force depend on the sign of $q_i q_j$. Positive $q_i q_j$ implies repulsion, and negative implies attraction.

Similarly, the equations of the charged particle system can also be reformulated in terms of an interaction graph representation. Since there are now two types of edges — attractive and repulsive — we consider the following adjacency tensor $\mathbf{z} \in \mathbb{R}^{N \times N \times 3}$, where $z_{ij0} = 1$ indicates an attractive edge, $z_{ij1} = 1$ indicates no edge, and $z_{ij2} = 1$ indicates a repulsive edge:

$$\dot{\mathbf{x}}_i = \sigma \cdot \left(-\sum_{i \neq j} z_{ij0} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|^3} + \sum_{i \neq j} z_{ij2} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|^3} \right), \tag{3}$$

where $q_i = q_j$ corresponds to $(z_{ij0}, z_{ij1}, z_{ij2}) = (0, 0, 1)$ and $q_i = -q_j$ corresponds to $(z_{ij0}, z_{ij1}, z_{ij2}) = (1, 0, 0)$.

The difference between the two dynamical systems lies in the probability distribution of edge types and the variation of interaction forces with distance. Based on the above analysis, we construct a parameterized dynamical system that includes three types of edges: attractive, null and repulsive, along with a parameter α that controls the relationship between interaction strength and distance.

Let z_{ij0}, z_{ij1}, and z_{ij2} denote the elements of the adjacency tensor z, where $z_{ij0} = 1$ for an attractive edge, $z_{ij1} = 1$ for a null edge, and $z_{ij2} = 1$ for a repulsive edge. The dynamical equations can then be written as follows:

$$\dot{\mathbf{x}}_i = \sigma \cdot \left(-\sum_{i \neq j} z_{ij0} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|^\alpha} + \sum_{i \neq j} z_{ij2} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|^\alpha} \right), \tag{4}$$

Assuming that the distribution of edges between all objects is identical, and each pair of objects has exactly one type of connection, the distribution of the adjacency tensor $\mathbf{z}$ will be determined by a distribution $\mathbf{p}$ over the set of edge types $\{0, 1, 2\}$.

For the spring system, since each pair of objects is connected or not connected with equal probability, this distribution is given by $\mathbf{p} = (1/2, 1/2, 0)$. For the charged particle system, since each particle has an equal probability of being positively or negatively charged, implying that each pair of particles attracts or

repels with equal probability, the distribution is given by $\mathbf{p} = (1/2, 0, 1/2)$. The corresponding α values for these systems are 0 and 3, respectively, indicating that the force is proportional to the distance for the spring system and inversely proportional to the square of the distance for the charged particle system.

Metadata can be generated from probability distributions over the parameters p and α. The parameter σ is adaptively adjusted to normalize the average velocity of the system, facilitating downstream training tasks. For simplicity, we use discrete distributions to construct the synthetic meta-dataset.

Specifically, we define the following parameter lists:

$$\texttt{list\_p} = \left[\left(\frac{1}{2}, \frac{1}{2}, 0\right), \left(\frac{1}{2}, 0, \frac{1}{2}\right), \left(\frac{5}{12}, \frac{5}{12}, \frac{1}{6}\right), \left(\frac{5}{12}, \frac{1}{6}, \frac{5}{12}\right), \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)\right]$$

$$\texttt{list\_alpha} = [-1, 0, 1, 2, 3, 4]$$

We generate sub-datasets by iterating over all combinations of these parameters. Each sub-dataset is then split into training, validation, and test sets of sizes 4000, 1000, and 1000 trajectories, respectively. To mitigate edge effects, we double the size of the datasets corresponding to $\alpha = -1$ and $\alpha = 4$.

All other simulation details follow those described in the original NRI paper, including trajectory lengths of 5000 time steps for the training and validation sets and 10000 time steps for the test set, with observations sampled every 100 time steps. Each sub-dataset contains four-dimensional features (2D position and 2D velocity) along with structural labels indicating the interaction graph.

These sub-datasets are then shuffled and combined to form a synthetic meta-dataset. To avoid data leakage, we exclude the target systems themselves — namely the spring and charged particle systems — as well as any datasets that are structurally similar to them. Specifically, we remove the configurations corresponding to $p = \left(\frac{1}{2}, \frac{1}{2}, 0\right), \left(\frac{5}{12}, \frac{5}{12}, \frac{1}{6}\right)$ with $\alpha = 0$ and $p = \left(\frac{1}{2}, 0, \frac{1}{2}\right), \left(\frac{5}{12}, \frac{1}{6}, \frac{5}{12}\right)$ with $\alpha = 3$.

1.2 Target Dataset

As previously mentioned, the target datasets for the spring system and the charged particle system can both be generated using their corresponding parameters. All other settings remain consistent with the NRI baseline, resulting in training, validation, and test datasets of sizes 50000, 10000, and 10000 trajectories, respectively. We verified the equivalence of our physical simulation routines with those used in the NRI baseline by plotting sample trajectories.

For the Kuramoto system, we directly utilized the relevant generation code from NRI and MPM to obtain the trajectories, employing identical settings. The sampling interval is 10 due to the faster dynamics of the Kuramoto system. The resulting dataset contains four-dimensional features: angular velocity, amplitude, phase, and natural frequency, along with structural labels indicating the interaction graph.

2 Implementation Details and Hyperparameters

2.1 Training Protocols

Algorithm 1 outlines the iterative training procedure of the proposed IPSI framework. This iterative scheme alternates between training a prior model for structural inference (SI_{prior}) and refining a joint model (SI_{joint}) under guidance from the prior. The training process can be initialized in two modes:

- **With prior (using synthetic meta-dataset):** SI_{prior} is pretrained on a collection of synthetic dynamical systems with known ground-truth structures. This yields an initial prior capable of producing soft edge distributions and node embeddings.
- **Without prior (baseline supervision):** When the synthetic meta-dataset is unavailable, an existing structure inference model (e.g., NRI) is used to generate pseudo-labels, and SI_{prior} is trained on these in a supervised fashion.

After the initialization step, the framework proceeds for R iteration rounds. In each round r, the following steps are performed:

1. $\mathrm{SI}_{\mathrm{prior}}$ is (re)trained on the current pseudo-labels $\mathcal{L}^{(r-1)}$ (except in the first round where pretrained weights are reused).

2. The trained prior model $\mathcal{F}^{(r)}_{\mathrm{prior}}$ produces a structural prior $q^{(r)}_{\mathrm{prior}}(z|x)$ and node-level embeddings $h^{(r)}$.

3. $\mathrm{SI}_{\mathrm{joint}}$ is trained on the target data using the inferred prior and embeddings as inputs. This model performs structure and dynamics inference in a unified manner.

4. The updated structural predictions from $\mathcal{F}^{(r)}_{\mathrm{joint}}$ are discretized to obtain new pseudo-labels $\mathcal{L}^{(r)}$, which are then fed into the next round.

This iterative loop allows $\mathrm{SI}_{\mathrm{joint}}$ to benefit from both learned dynamics representations and progressively refined structural priors. As shown in our experiments, this iterative process significantly improves the structural inference accuracy, even under limited supervision.

Algorithm 1 Iterative Training Procedure of IPSI

Require: Target dataset $\mathcal{D}$, synthetic meta-dataset $\mathcal{D}_{\mathrm{meta}}$ (optional), baseline model $\mathcal{M}_{\mathrm{base}}$ (optional), number of iteration rounds R, boolean *with_prior*
Ensure: Trained $\mathrm{SI}_{\mathrm{prior}}$ and $\mathrm{SI}_{\mathrm{joint}}$ modules

Initialize parameters of $\mathrm{SI}_{\mathrm{prior}}$ and $\mathrm{SI}_{\mathrm{joint}}$
if *with_prior* **then** ▷ (Step 0) Supervision from synthetic meta-dataset
    Train SI_{prior} on $\mathcal{D}_{\mathrm{meta}}$ using ground-truth labels
else ▷ (Step 0) Supervision via baseline model
    Train baseline model $\mathcal{M}_{\mathrm{base}}$ (e.g., NRI) on $\mathcal{D}$
    $\mathcal{L}^{(0)} \leftarrow \mathrm{InferEdges}(\mathcal{M}_{\mathrm{base}}, \mathcal{D})$
    Train SI_{prior} on $\mathcal{D}$ using $\mathcal{L}^{(0)}$
end if
for round $r = 1$ to R **do**
    ▷ (Step 1) Train $\mathrm{SI}_{\mathrm{prior}}$ with updated pseudo-labels
    if $r > 1$ **then**
        $\mathcal{F}^{(r)}_{\mathrm{prior}} \leftarrow \mathrm{Train}(SI_{\mathrm{prior}}, \mathcal{D}, \mathcal{L}^{(r-1)})$
    else
        $\mathcal{F}^{(1)}_{\mathrm{prior}} \leftarrow SI_{\mathrm{prior}}$ ▷ First round uses model trained in Step 0
    end if
    ▷ (Step 2) Generate prior and embeddings
    $q^{(r)}_{\mathrm{prior}}(z|x) \leftarrow \mathrm{InferPrior}(\mathcal{F}^{(r)}_{\mathrm{prior}}, \mathcal{D})$
    $h^{(r)} \leftarrow \mathrm{GenerateEmbedding}(\mathcal{F}^{(r)}_{\mathrm{prior}}, \mathcal{D})$
    ▷ (Step 3) Train $\mathrm{SI}_{\mathrm{joint}}$ with prior guidance
    $\mathcal{F}^{(r)}_{\mathrm{joint}} \leftarrow \mathrm{Train}(SI_{\mathrm{joint}}, \mathcal{D}, q^{(r)}_{\mathrm{prior}}(z|x), h^{(r)})$
    ▷ (Step 4) Update pseudo-labels from $\mathrm{SI}_{\mathrm{joint}}$ outputs
    $\mathcal{L}^{(r)} \leftarrow \mathrm{InferEdges}(\mathcal{F}^{(r)}_{\mathrm{joint}}, \mathcal{D})$
end for
return Final models $\mathcal{F}^{(R)}_{\mathrm{prior}}$, $\mathcal{F}^{(R)}_{\mathrm{joint}}$

2.2 Training Details

This section provides detailed training configurations for each stage of the IPSI framework. We separately describe the settings used for (1) training the prior model on the synthetic meta-dataset, (2) iterative training of $\mathrm{SI}_{\mathrm{prior}}$ on the target data, and (3) training $\mathrm{SI}_{\mathrm{joint}}$ under prior supervision.

2.2.1 $\mathrm{SI}_{\mathrm{prior}}$ Pretraining on Meta-Dataset

When structural prior supervision is available, we initialize the $\mathrm{SI}_{\mathrm{prior}}$ module by pretraining it on the synthetic meta-dataset $\mathcal{D}_{\mathrm{meta}}$. This dataset is constructed from a diverse family of parameterized dynamical systems, and covers a broad range of force functions, edge types, and interaction patterns.

The pretraining process optimizes a supervised edge classification objective using cross-entropy loss, where the model learns to predict discrete edge types between all node pairs based on their observed dynamics. We use the Adam optimizer with a learning rate of `5e-4` and weight decay of `1e-6`. Training is conducted for `200` epochs with a batch size of `256`, the modules is configured with a hidden dimension of `256` and dropout rate set to `0.2` to mitigate overfitting.

This pretraining phase yields a model capable of producing meaningful structural predictions even when applied to previously unseen systems, providing a useful initialization for the subsequent iterative training on the target data.

2.2.2 Training of $\mathrm{SI}_{\mathrm{prior}}$ via pseudo-labels

$\mathrm{SI}_{\mathrm{prior}}$ module is also trained on the target dataset $\mathcal{D}$ using pseudo-labels as supervision. This training procedure applies to both the following scenarios: (1) when no structural prior is available, i.e., in the absence of synthetic meta-dataset supervision, $\mathrm{SI}_{\mathrm{prior}}$ is initially trained on pseudo-labels generated by an external baseline model such as NRI; and (2) in all subsequent iteration rounds (i.e., $r \geq 2$) regardless of the initialization strategy, where updated pseudo-labels are provided by the $\mathrm{SI}_{\mathrm{joint}}$ module trained in the previous round.

At the beginning of each iteration round r, the model is trained to minimize cross-entropy loss over discrete edge labels $\mathcal{L}^{(r-1)}$. The training is performed using the Adam optimizer with a learning rate of `5e-4`. We use a batch size of `128` and train the model for `50` epochs per round. The weights are reinitialized in each iteration to continuously escape local optima. The weight decay of `1e-6` and dropout rate of `0.2` are also applied.

2.2.3 Training of $\mathrm{SI}_{\mathrm{joint}}$ with Prior Guidance

The majority of training settings for $\mathrm{SI}_{\mathrm{joint}}$ are kept consistent with the NRI baseline to ensure fair comparison. Specifically, we adopt the Adam optimizer with a learning rate of `5e-4`, a batch size of `128`, and a KL weighting coefficient β set to `1.0`. The Gumbel-Softmax temperature for edge sampling is fixed at `0.5` throughout training. And a learning rate decay schedule that halves the learning rate every `200` epochs is applied.

The number of iteration rounds is set to `2` by default, and $\mathrm{SI}_{\mathrm{joint}}$ is trained for `500` epochs per round.

2.2.4 Additional Settings for the Kuramoto System and Trajectory Embedding.

For the Kuramoto system, we follow the same training protocol as in the baseline implementation to ensure a fair comparison. The input feature dimension is modified from 4 (used in the spring and charged particle systems) to 3. Additionally, we apply a hard constraint on edge type classification by fixing the first edge tpye (corresponding to null interactions) as always inactive.

We adopt a 1D convolutional embedding module (1D-CNN) for the Kuramoto system, consistent with the baseline. More broadly, we investigate the impact of trajectory embedding modules on performance. We find that the choice of trajectory embedding modules for $\mathrm{SI}_{\mathrm{prior}}$ has limited influence on performance in most settings; however, pretraining on the synthetic meta-dataset significantly benefits from the use of GRU embedding module. As a result, we standardize the embedding module choice by using a GRU module for $\mathrm{SI}_{\mathrm{prior}}$ across all systems. For $\mathrm{SI}_{\mathrm{joint}}$, we preserve the same embedding module type as in the original baseline for each system to ensure fair comparison: MLP for the spring system, and 1D-CNN for both the charged particle and Kuramoto systems.

2.3 Compute Resources

All experiments were conducted on a single NVIDIA RTX 3090 Ti GPU with 24GB memory using PyTorch 2.6.0 and CUDA 12.6. Each complete training run of IPSI, including all $R = 5$ iteration rounds, takes

Table 1: MSE over 20 steps with 5 interacting objects. For LSTM, NRI, SUGAR, and MPM, we directly used the results from the original papers after checking the consistency of the benchmark. For iSIDG, we used our own measured results because different datasets generation methods were used.

Model	**Springs**			**Charged**			**Kuramoto**		
	1	10	20	1	10	20	1	10	20
LSTM	4.13e-8	2.19e-5	7.02e-4	1.68e-3	6.45e-3	1.49e-2	**3.44e-4**	**1.29e-2**	4.74e-2
NRI	3.12e-8	3.29e-6	2.13e-5	1.05e-3	3.21e-3	7.06e-3	1.40e-2	2.01e-2	3.26e-2
SUGAR	3.71e-8	3.86e-6	1.53e-5	1.18e-3	3.43e-3	7.38e-3	2.12e-2	9.45e-2	1.83e-1
MPM	8.89e-9	5.99e-7	2.52e-6	**7.29e-4**	2.57e-3	**5.41e-3**	1.57e-2	2.73e-2	5.36e-2
iSIDG	3.13e-8	3.24e-6	2.11e-5	1.06e-3	3.01e-3	7.01e-3	1.35e-2	2.11e-2	3.36e-2
IPSI (NRI-based)	3.08e-8	3.28e-6	2.16e-5	9.67e-4	3.14e-3	6.91e-3	1.36e-2	1.99e-2	**3.24e-2**
IPSI (MPM-based)	**8.81e-9**	**5.98e-7**	**2.48e-6**	7.32e-4	**2.56e-3**	5.44e-3	1.61e-2	2.69e-2	5.32e-2

approximately 5–8 hours for the 5-object systems and 20–30 hours for the 10-object systems. Within each round, training $\mathrm{SI}_{\mathrm{prior}}$ takes roughly 10–20 minutes, while training $\mathrm{SI}_{\mathrm{joint}}$ takes 1–3 hours depending on the system and number of objects. Due to the training strategy of iterations, IPSI requires 2–3 times more training time compared to the baseline model. This trade-off results in more accurate structure predictions, which significantly outperform the performance of the baseline model even when the latter is trained for the same total duration. All experiments were run sequentially without multi-GPU parallelism, though our implementation supports data parallelism via standard PyTorch mechanisms. Training scripts and environment settings will be released to ensure reproducibility of all results.

3 Results about Trajectory Prediction Errors

To complement our evaluation based on structure inference accuracy, we report the mean squared error (MSE) of trajectory prediction across different models and datasets in Table 1. This metric evaluates how accurately each model can forecast future object positions based on the inferred interaction graph. We observe that IPSI achieves comparable trajectory prediction performance under both the prior-supervised and prior-free settings, indicating that the presence of structural priors has limited impact on dynamics prediction once the iterative refinement process is applied. As such, we report only the results from the prior-free setting to avoid redundancy. The testing protocol strictly follows the baseline setup: models are evaluated on a test set containing longer trajectories of 100 time steps. The first 49 time steps are used to infer the interaction graph, which is then held fixed to predict the next 20 steps. The predicted trajectories are compared against ground-truth trajectories using mean squared error. This ensures that all methods are assessed under the same conditions and that structural inference quality is evaluated in terms of its impact on future prediction.

While IPSI consistently improves the structural accuracy of inferred graphs over baseline models, the overall improvements in MSE across models and datasets are relatively small and not always consistent with gains in edge accuracy. This discrepancy is not coincidental, but rather stems from the nature of the baseline training objective. Specifically, baseline models such as NRI perform structure inference by minimizing trajectory prediction error alone, without access to ground-truth edge labels. As a result, the edges that are incorrectly inferred by such models tend to be those whose misclassification has negligible impact on the resulting dynamics.

In other words, trajectory-prediction-driven models implicitly learn to prioritize edges that are important for accurately predicting motion, while tolerating errors in edges whose dynamical contribution is minor. For instance, in spring systems with dense connectivity, removing a weak spring connection may have little effect due to redundancy in neighboring forces. In charged particle systems, omitting an interaction between distant particles may be inconsequential because the Coulomb force decays quadratically with distance.

This explains why IPSI, despite achieving significantly higher structural accuracy—including correcting many of the weak or long-range edges missed by baseline models—does not always yield substantial gains in trajectory prediction. The additional edges it correctly infers are precisely those that contribute little to the prediction loss, and thus were deprioritized.

4 Discussion on Alternating Updates vs. Joint Minimization

We provide additional clarification and empirical evidence regarding our choice of the alternating update strategy between $\mathrm{SI}_{\mathrm{prior}}$ and $\mathrm{SI}_{\mathrm{joint}}$. We implemented a variant of our framework that performs *joint minimization*—that is, both $\mathrm{SI}_{\mathrm{prior}}$ and $\mathrm{SI}_{\mathrm{joint}}$ are updated within the same epoch, instead of adopting full alternating updates.

Rationale for Alternating Updates. Our design choice is motivated by two considerations:

1. **Design motivation:** We hypothesize that more accurate structural priors estimated by $\mathrm{SI}_{\mathrm{prior}}$ enhance $\mathrm{SI}_{\mathrm{joint}}$, and conversely, refined inference from $\mathrm{SI}_{\mathrm{joint}}$ further benefits $\mathrm{SI}_{\mathrm{prior}}$. Alternating full updates naturally supports this mutual refinement process.
2. **Stability:** As $\mathrm{SI}_{\mathrm{joint}}$ receives embeddings generated by $\mathrm{SI}_{\mathrm{prior}}$, it is desirable to ensure a stable input distribution. Updating both modules simultaneously within each epoch may introduce rapidly shifting inputs, which could hinder convergence stability.

Empirical Comparison. To examine this effect, we trained a variant adopting the joint minimization strategy. The number of epochs and learning rate schedules were adjusted to ensure a fair comparison. Experiments were conducted on the 15-node Gene Regulatory Network dataset from DoSI. As shown in Table 2, while the joint-minimization variant still outperforms the baseline NRI, it remains inferior to our alternating update design.

Table 2: Comparison of AUROC performance between alternating and joint update strategies on the DoSI dataset (15-node Gene Regulatory Network).

Model	**NRI**	**PreBoot-SI (alternating)**	**PreBoot-SI (joint minimization)**
AUROC (%)	78.08	**84.40**	81.89

These results support our design rationale: alternating updates yield better stability and overall performance, validating the architectural choice in IPSI.

5 Discussion on the Choice of Node Embedding Architecture

We elaborate on our design choice of using a GRU-based node embedding rather than an MLP, particularly within the $\mathrm{SI}_{\mathrm{prior}}$ module trained on the synthetic meta-dataset.

Motivation for Using GRU. The motivation arises from the nature of the pretraining task. The meta-dataset is constructed to capture recurring trajectory–structure patterns across systems exhibiting similar dynamics. For instance, if two objects consistently move away from each other over time, this behavior likely corresponds to a repulsive interaction—a temporal pattern that recurrent units such as GRUs are particularly suited to model. Therefore, employing GRU embeddings allows $\mathrm{SI}_{\mathrm{prior}}$ to more effectively encode temporal dependencies that underlie the structural prior learning process.

Isolating Architectural Effects. To evaluate the architectural effect independently of the IPSI mechanism, we also replaced the MLP-based node embeddings in the original NRI baseline with GRUs. This modification produced only marginal performance differences, suggesting that the performance gain of IPSI stems primarily from its iterative structural inference design rather than the specific embedding architecture.

Empirical Results. Table 3 summarizes the comparative results on the Charged dataset (5-node configuration). When using GRU embeddings, we observe slightly improved performance, especially in the dynamic-prior setting where the synthetic meta-dataset is utilized. This aligns with our hypothesis that

Table 3: Comparison between MLP-based and GRU-based node embeddings on the Charged dataset (5 nodes).

Model Type	**Node Embedding**	**Edge Prediction Accuracy (%)**
NRI	GRU	81.9
NRI	MLP	82.2
IPSI (w/o prior)	GRU	89.3
IPSI (w/o prior)	MLP	88.9
IPSI (w/ prior)	GRU	**91.2**
IPSI (w/ prior)	MLP	89.1

temporal encoders benefit structural prior estimation, while their effect remains secondary compared to the overall IPSI framework design.

These observations indicate that while recurrent encoders such as GRU offer a modest advantage in capturing temporal dependencies, the main improvements in IPSI performance originate from the iterative pretraining and structural refinement mechanism rather than from the node embedding architecture itself.