A IMPLEMENTATION DETAILS

A.1 NOTATION

Table 1: Notation used throughout this work

Variable	Description
$\mathbf{p}_i$	Position of node i
$\mathbf{u}_i$	Velocity of node i
$\mathbf{x}_i$	State (Position & Velocity) of node i
$\mathbf{v}_i$	Augmented State (Position & Velocity & Orientation) of node i
$\mathbf{h}_i$	Hidden vector of node i
$\square_{j\|i}$	Variable $\square$ of node j expressed at local coordinate frame of node i
$\mathbf{f}_i$	External force exerted at node i
f_v, f_e, g_v	MLPs
$[\cdot, \cdot]$	Vector concatenation (along the feature dimension)

A.2 AETHER

Here we present the full Aether architecture. We first describe the details of the neural field used for field discovery, and then we describe our graph network formulated as a variational autoencoder (Kingma & Welling, 2014; Rezende et al., 2014).

A.2.1 NEURAL FIELD

In its general form, the neural field takes as inputs query positions $\mathbf{p} \in \mathbb{R}^2$ and orientations $\theta \in \mathcal{S}^1$, as well as a latent code $\mathbf{z} \in \mathbb{R}^{D_z}$ used to condition the field, and predicts latent forces at the query positions-orientations. Thus, it is defined as $\mathbf{f} : \mathbb{R}^2 \times \mathcal{S}^1 \times \mathbb{R}^{D_z} \to \mathbb{R}^2$. Depending on the task at hand, we can omit the latent code $\mathbf{z}$, *e.g.* if we are modelling a static field, or the orientations θ, if we have prior knowledge that the field is independent to them.

Encoding positions We encode the input positions using Gaussian random Fourier features (Tancik et al., 2020), $\gamma(\mathbf{p}) = [\cos(2\pi\mathbf{B}\mathbf{p}), \sin(2\pi\mathbf{B}\mathbf{p})]^\top$, where $\mathbf{B} \in \mathbb{R}^{\frac{D_c}{2} \times 2}$ is a matrix with entries sampled from a Gaussian distribution, $\mathbf{B}_{kl} \sim \mathcal{N}(0, \sigma^2)$. Throughout the experiments, we use a unit variance $\sigma^2 = 1$, and $\frac{D_c}{2} = 256$. Thus, the encoded positions have a dimension of 512.

Encoding orientations For the orientations θ, we follow Kofinas et al. (2021), and use the angles of the velocity vectors as a proxy. We represent orientations as unit vectors $\boldsymbol{\theta} = [\cos\theta, \sin\theta]^\top$, and encode them with a linear layer $\delta(\boldsymbol{\theta}) = \mathbf{W}_\omega \boldsymbol{\theta}$, where $\mathbf{W}_\omega \in \mathbb{R}^{D_c \times 2}$. Finally, we concatenate the encoded positions and orientations in a single vector before we feed them as input to the neural field.

Latent code The latent code $\mathbf{z}$ "summarizes" the input graph such that it isolates global field effects. We employ a simple global spatio-temporal attention mechanism, similar to Li et al. (2016), that aggregates the input system in a latent vector representation. First, we define object embeddings $\mathbf{o}_i = \mathrm{GRU}\big(\mathbf{W}_g \mathbf{x}_i^{1:T}\big)$, where $\mathbf{W}_g \in \mathbb{R}^{D_\mathrm{o} \times D_\mathrm{in}}$ is a matrix used to linearly transform the inputs, and GRU is the Gated Recurrent Unit (Cho et al., 2014). We also define temporal embeddings $\mathbf{t} = \mathrm{PE}(t)$, where PE are positional encodings (Vaswani et al., 2017), defined as:

$$\mathrm{PE}(t)_{2i} = \sin\Big(t/10000^{2i/D_\mathrm{s}}\Big), \tag{13}$$

$$\mathrm{PE}(t)_{2i+1} = \cos\Big(t/10000^{2i/D_\mathrm{s}}\Big), \tag{14}$$

where i is the i-th dimension.

The aggregation is then defined as follows:

$$\mathbf{z} = \sum_{i,t} \mathrm{softmax}\big(f_a(\mathbf{s}_i^t)\big) \cdot f_b(\mathbf{s}_i^t), \quad \text{with} \quad \mathbf{s}_i^t = \big[\mathbf{x}_i^t, \mathbf{o}_i\big] + \mathbf{t}, \tag{15}$$

where $f_a : \mathbb{R}^{D_\text{s}} \to \mathbb{R}$, $f_b : \mathbb{R}^{D_\text{s}} \to \mathbb{R}^{D_\text{z}}$ are 2-layer MLPs with SiLU activations Ramachandran et al. (2018) in between. They can be summarized as:

$$f_a \coloneqq \{\text{Linear}(D_\text{s}, D_\text{z}) \to \text{SiLU} \to \text{Linear}(D_\text{z}, 1)\}, \tag{16}$$

$$f_b \coloneqq \{\text{Linear}(D_\text{s}, D_\text{z}) \to \text{SiLU} \to \text{Linear}(D_\text{z}, D_\text{z})\}. \tag{17}$$

In all experiments, we use $D_\text{o} = 512, D_\text{s} = D_\text{o} + D_\text{in} = 516, D_\text{z} = 512$.

Neural field conditioning We condition the neural field using FiLM (Perez et al., 2018). Following the implementation details of FiLM, in practice, we use the following equation for a FiLM layer:

$$\mathbf{h}' \coloneqq \text{FiLM}(\mathbf{h}, \mathbf{z}) = (1 + \alpha(\mathbf{z})) \odot \mathbf{h} + \beta(\mathbf{z}), \tag{18}$$

where $\mathbf{h}$ is the encoded input in the first FiLM layer, or the conditioned input in subsequent FiLM layers, and $\alpha : \mathbb{R}^{D_\text{z}} \to \mathbb{R}^{D_\text{h}}, \beta : \mathbb{R}^{D_\text{z}} \to \mathbb{R}^{D_\text{h}}$ are MLPs. This equation deviates slightly from section 2.3, since it predicts the residual of a multiplicative modulation. This approach can be beneficial during the early stages of training, since it initially defaults to an identity transformation for zero-initialized weights, while the alternative can "zero out" the network outputs. For both α and β we use 2-layer MLPs with SiLU activations in-between.

$$\alpha \coloneqq \{\text{Linear}(D_\text{z}, D_\text{h}) \to \text{SiLU} \to \text{Linear}(D_\text{h}, D_\text{h})\} \tag{19}$$

$$\beta \coloneqq \{\text{Linear}(D_\text{z}, D_\text{h}) \to \text{SiLU} \to \text{Linear}(D_\text{h}, D_\text{h})\}. \tag{20}$$

In all experiments, we use $D_\text{h} = 512$.

Full neural field The full neural field is a 3-layer MLP with SiLU (Ramachandran et al., 2018) activations in-between, and FiLM layers after the first two linear layers, and outputs a latent force field. The neural field can be summarized as

$$\mathbf{f}(\mathbf{p}, \boldsymbol{\theta}, \mathbf{z}) = \{\text{Linear} \to \text{FiLM} \to \text{SiLU} \to \text{Linear} \to \text{FiLM} \to \text{SiLU} \to \text{Linear}\} \tag{21}$$

A.2.2 AETHER AS A VARIATIONAL AUTOENCODER

Here we present our graph network architecture that closely follows Graber & Schwing (2020); Kofinas et al. (2021). The model is formulated as a variational autoencoder (Kingma & Welling, 2014; Rezende et al., 2014) with latent edge types that infers a latent graph structure. The encoder is tasked with predicting interactions between object pairs, while the decoder uses the sampled graph structure to make predictions. As mentioned in section 3.3, our full architecture also integrates G-LoCS, *i.e.* the augmented node states include the predicted forces exerted at the target node, as well as the state of the auxiliary origin-node, expressed in the local frame of the target node.

Encoder Equations (22) to (24) describe the message passing steps of our graph network. In these equations, we process each timestep independently.Then, in eqs. (25) and (26) we compute the evolution of edge embeddings over time with LSTMs (Hochreiter & Schmidhuber, 1997), and in eqs. (27) and (28) we estimate the posterior and the learned prior over our edges.

$$\mathbf{h}_{j,i}^{(1),t} = f_e^{(1)}\left(\left[\mathbf{v}_{j|i}^t, \mathbf{f}_{j|i}^t, \mathbf{v}_{i|i}^t, \mathbf{f}_{i|i}^t, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) \tag{22}$$

$$\mathbf{h}_{i}^{(1),t} = f_v^{(1)}\left(g_v^{(1)}\left(\left[\mathbf{v}_{i|i}^t, \mathbf{f}_{i|i}^t, \mathbf{v}_{\mathcal{O}|i}^t\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j \in \mathcal{N}(i)} \mathbf{h}_{j,i}^{(1),t}\right) \tag{23}$$

$$\mathbf{h}_{j,i}^{(2),t} = f_e^{(2)}\left(\left[\mathbf{h}_{i}^{(1),t}, \mathbf{h}_{j,i}^{(1),t}, \mathbf{h}_{j}^{(1),t}\right]\right) \tag{24}$$

$$\mathbf{h}_{(j,i),\text{prior}}^{t} = \text{LSTM}_\text{prior}\left(\mathbf{h}_{j,i}^{(2),t}, \mathbf{h}_{(j,i),\text{prior}}^{t-1}\right) \tag{25}$$

$$\mathbf{h}_{(j,i),\text{enc}}^{t} = \text{LSTM}_\text{enc}\left(\mathbf{h}_{j,i}^{(2),t}, \mathbf{h}_{(j,i),\text{enc}}^{t+1}\right) \tag{26}$$

$$p_\phi\left(\mathbf{z}^t | \mathbf{x}^{1:t}, \mathbf{z}^{1:t-1}\right) = \text{softmax}\left(f_\text{prior}\left(\mathbf{h}_{(j,i),\text{prior}}^{t}\right)\right) \tag{27}$$

$$q_\phi\left(\mathbf{z}_{j,i}^t | \mathbf{x}\right) = \text{softmax}\left(f_\text{enc}\left(\left[\mathbf{h}_{(j,i),\text{prior}}^{t}, \mathbf{h}_{(j,i),\text{enc}}^{t}\right]\right)\right) \tag{28}$$

The functions $f_e^{(1)}, f_v^{(1)}, f_e^{(2)}, g_v^{(1)}, f_\text{prior}, f_\text{enc}$ denote MLPs.

Decoder The decoder samples $\mathbf{z}^t_{(j,i)}$ using Gumbel-Softmax (Maddison et al., 2017; Jang et al., 2017). The following equations formalize a message passing scheme performed for the current timestep, and another one performed for the hidden node states. Both are used to update the hidden node states, and to make predictions for the next timestep. As mentioned in section 2.2, we make predictions in the local coordinate frame of each node. Thus, we perform an inverse transformation for each node to transform the predictions back to the global coordinate frame.

$$\mathbf{m}^t_{j,i} = \sum_k z^t_{(j,i),k} f^k\left(\left[\mathbf{v}^t_{j|i}, \mathbf{f}^t_{j|i}, \mathbf{v}^t_{i|i}, \mathbf{f}^t_{i|i}, \mathbf{v}^t_{\mathcal{O}|i}\right]\right) \tag{29}$$

$$\mathbf{m}^t_i = f^{(3)}_v\left(g^{(3)}_v\left(\left[\mathbf{v}^t_{i|i}, \mathbf{f}^t_{i|i}, \mathbf{v}^t_{\mathcal{O}|i}\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{m}^t_{j,i}\right) \tag{30}$$

$$\mathbf{h}^t_{j,i} = \sum_k z^t_{(j,i),k} g^k\left(\left[\mathbf{h}^t_j, \mathbf{h}^t_i\right]\right) \tag{31}$$

$$\mathbf{n}^t_i = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{h}^t_{(j,i)} \tag{32}$$

$$\mathbf{h}^{t+1}_i = \text{GRU}\left(\left[\mathbf{n}^t_i, \mathbf{m_i}^t\right], \mathbf{h}^t_i\right) \tag{33}$$

$$\boldsymbol{\mu}^{t+1}_i = \mathbf{x}^t_i + \mathbf{R}^t_i \cdot f^{(4)}_v\left(\mathbf{h}^{t+1}_i\right) \tag{34}$$

$$p(\mathbf{x}^{t+1}_i | \mathbf{x}^{1:t}, \mathbf{z}^{1:t}) = \mathcal{N}\left(\boldsymbol{\mu}^{t+1}_i, \sigma^2\mathbf{I}\right) \tag{35}$$

A.2.3 G-LoCS

Encoder

$$\mathbf{h}^{(1),t}_{j,i} = f^{(1)}_e\left(\left[\mathbf{v}^t_{j|i}, \mathbf{v}^t_{i|i}, \mathbf{v}^t_{\mathcal{O}|i}\right]\right) \tag{36}$$

$$\mathbf{h}^{(1),t}_i = f^{(1)}_v\left(g^{(1)}_v\left(\left[\mathbf{v}^t_{i|i}, \mathbf{v}^t_{\mathcal{O}|i}\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{h}^{(1),t}_{j,i}\right) \tag{37}$$

$$\mathbf{h}^{(2)}_{j,i} = f^{(2)}_e\left(\left[\mathbf{h}^{(1),t}_i, \mathbf{h}^{(1),t}_{j,i}, \mathbf{h}^{(1),t}_j\right]\right) \tag{38}$$

$$\mathbf{h}^t_{(j,i),\text{prior}} = \text{LSTM}_{\text{prior}}\left(\mathbf{h}^{(2),t}_{j,i}, \mathbf{h}^{t-1}_{(j,i),\text{prior}}\right) \tag{39}$$

$$\mathbf{h}^t_{(j,i),\text{enc}} = \text{LSTM}_{\text{enc}}\left(\mathbf{h}^{(2),t}_{j,i}, \mathbf{h}^{t+1}_{(j,i),\text{enc}}\right) \tag{40}$$

$$p_\phi\left(\mathbf{z}^t | \mathbf{x}^{1:t}, \mathbf{z}^{1:t-1}\right) = \text{softmax}\left(f_{\text{prior}}\left(\mathbf{h}^t_{(j,i),\text{prior}}\right)\right) \tag{41}$$

$$q_\phi\left(\mathbf{z}^t_{j,i} | \mathbf{x}\right) = \text{softmax}\left(f_{\text{enc}}\left(\left[\mathbf{h}^t_{(j,i),\text{prior}}, \mathbf{h}^t_{(j,i),\text{enc}}\right]\right)\right) \tag{42}$$

Decoder

$$\mathbf{m}^t_{j,i} = \sum_k z^t_{(j,i),k} f^k\left(\left[\mathbf{v}^t_{j|i}, \mathbf{v}^t_{i|i}, \mathbf{v}^t_{\mathcal{O}|i}\right]\right) \tag{43}$$

$$\mathbf{m}^t_i = f^{(3)}_v\left(g^{(3)}_v\left(\left[\mathbf{v}^t_{i|i}, \mathbf{v}^t_{\mathcal{O}|i}\right]\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{m}^t_{j,i}\right) \tag{44}$$

$$\mathbf{h}^t_{j,i} = \sum_k z^t_{(j,i),k} g^k\left(\left[\mathbf{h}^t_j, \mathbf{h}^t_i\right]\right) \tag{45}$$

$$\mathbf{n}^t_i = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{h}^t_{(j,i)} \tag{46}$$

$$\mathbf{h}^{t+1}_i = \text{GRU}\left(\left[\mathbf{n}^t_i, \mathbf{m_i}^t\right], \mathbf{h}^t_i\right) \tag{47}$$

$$\boldsymbol{\mu}^{t+1}_i = \mathbf{x}^t_i + \mathbf{R}^t_i \cdot f^{(4)}_v\left(\mathbf{h}^{t+1}_i\right) \tag{48}$$

$$p(\mathbf{x}^{t+1}_i | \mathbf{x}^{1:t}, \mathbf{z}^{1:t}) = \mathcal{N}\left(\boldsymbol{\mu}^{t+1}_i, \sigma^2\mathbf{I}\right) \tag{49}$$

Training Our full VAE model is trained by minimizing the negative Evidence Lower Bound (ELBO), which comprises the reconstruction loss of the predicted trajectories (positions and velocities) and the KL divergence.

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\phi(\mathbf{z}|\mathbf{x})] \tag{50}$$

Following Graber & Schwing (2020), the reconstruction loss and the KL divergence take the following form:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] = -\sum_i \sum_t \frac{||\mathbf{x}_i^t - \boldsymbol{\mu}_i^t||}{2\sigma^2} + \frac{1}{2}\log\left(2\pi\sigma^2\right), \tag{51}$$

$$\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\phi(\mathbf{z}|\mathbf{x})] = \sum_{t=1}^{T}\left(\mathbb{H}(q_\phi(\mathbf{z}_{ji}^t|\mathbf{x})) - \sum_{\mathbf{z}_{ji}^t} q_\phi(\mathbf{z}_{ji}^t|\mathbf{x}) \log p_\phi(\mathbf{z}_{ji}^t|\mathbf{x}^{1:t}, \mathbf{z}^{1:t-1})\right), \tag{52}$$

where $\mathbb{H}$ denotes the entropy operator. In all experiments, we set the variance $\sigma^2 = 10^{-5}$.

We train Aether using Adam (Kingma & Welling, 2014). In all experiments, we use a learning rate of 5e$-$4.

A.2.4 Source oracle

The *source oracle* modifies LoCS (Kofinas et al., 2021) to use virtual nodes. Since graph networks are permutation invariant, we cannot just include the sources as nodes of the graph and perform message passing, as the network would not be able to distinguish particles from sources. Thus, we treat the sources separately in the message passing so that the network can identify them. More specifically, we introduce a new message function that computes field source $\rightarrow$ particle messages. Furthermore, we introduce a separate aggregation function in the update step that only aggregates the messages from field sources. We denote the set of field sources as $\mathcal{S}$. The state of a field source $s \in \mathcal{S}$ is denoted as $\mathbf{v}_s$, while the same state expressed in the local coordinate frame of node i is denoted as $\mathbf{v}_{s|i}$. The source oracle graph network is defined as follows:

$$\mathbf{h}_{j,i}^t = f_e\left(\left[\mathbf{v}_{j|i}^t, \mathbf{v}_{i|i}^t\right]\right), \tag{53}$$

$$\mathbf{h}_{s,i}^t = f_s\left(\left[\mathbf{v}_{s|i}^t, \mathbf{v}_{i|i}^t\right]\right), \tag{54}$$

$$\boldsymbol{\Delta}\mathbf{x}_{i|i}^{t+1} = f_v\left(g_v\left(\mathbf{v}_{i|i}^t\right) + \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{h}_{j,i}^t + \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\mathbf{h}_{s,i}^t\right), \tag{55}$$

where f_s is an MLP.

A.3 Computing resources

All experiments were performed on single GPUs. We used 2 different GPU models, namely the Nvidia RTX 2080 Ti, and Nvidia GTX 1080 Ti. Our source code was written in PyTorch (Paszke et al., 2019), version 1.4.0, and CUDA 10.0.

B Dataset details

B.1 Charged Particles

Kipf et al. (2018) introduced a dataset of interacting charged particles. Charged particles interact via electrostatic Coulomb forces. We assume a set of N particles, and each particle has a position $\mathbf{p}_i^t \in \mathbb{R}^D$ and a charge $q_i \in \mathbb{R}$. The force $\mathbf{f}_{j,i}^t$ exerted from particle j to particle i is computed as

follows:

$$\mathbf{p}_{j,i}^t = \mathbf{p}_j^t - \mathbf{p}_i^t \tag{56}$$

$$\hat{\mathbf{p}}_{j,i}^t = \frac{\mathbf{p}_{j,i}^t}{\left\|\mathbf{p}_{j,i}^t\right\|} \tag{57}$$

$$\mathbf{f}_{j,i}^t = C \cdot q_i q_j \frac{\hat{\mathbf{p}}_{j,i}^t}{\left\|\mathbf{p}_{j,i}^t\right\|^2} \tag{58}$$

Since forces only depend on positions at the current timestep, in the following equations, we omit the time indices to reduce clutter. The total force exerted at particle i is

$$\mathbf{f}_i = \sum_{j=1, j\neq i}^{N} \mathbf{f}_{j,i} = C \cdot q_i \sum_{j=1, j\neq i}^{N} q_j \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2} \tag{59}$$

The electric field is a vector field, whose value at the test position $\mathbf{p}_i$ assumes a positive test charge $q_i = 1$, and is defined as:

$$\mathbf{E}_{j,i} = \frac{\mathbf{f}_{j,i}}{q_i} = C \cdot q_j \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2} \tag{60}$$

$$\mathbf{E}_i = \sum_{j=1, j\neq i}^{N} \mathbf{E}_{j,i} = C \cdot \sum_{j=1, j\neq i}^{N} q_j \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2} \tag{61}$$

Our first experiment aims to study the effect of static fields, *i.e.* a single field across all train, validation, and test simulations. We extend the charged particles dataset by adding a number of immovable sources. Overall, these sources act like regular particles, exerting forces on the observable particles, except we ignore any forces exerted to them, and fix their positions and velocities to zero. We use $N = 5$ "observable" particles and $M = 20$ "source" particles. In all experiments, we assume unit charges, $q_i = \pm 1$, and $C = 1$. The probabilities of positive or negative charges are equal. Then, the forces and the electric field can be simplified as:

$$\mathbf{f}_{j,i} = \text{sign}(q_i q_j) \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2} \tag{62}$$

$$\mathbf{E}_i = \sum_{j=1, j\neq i}^{N} \text{sign}(q_j) \frac{\hat{\mathbf{p}}_{j,i}}{\left\|\mathbf{p}_{j,i}\right\|^2} \tag{63}$$

The net force exerted at a particle $i \in \{1, \ldots, N\}$ is computed as

$$\mathbf{f}_i = \sum_{j=1, j\neq i}^{N+M} \mathbf{f}_{j,i} = \underbrace{\sum_{j=1, j\neq i}^{N} \mathbf{f}_{j,i}}_{\text{particles}} + \underbrace{\sum_{j=N+1}^{N+M} \mathbf{f}_{j,i}}_{\text{field}} \tag{64}$$

We generate a dataset of 50,000 simulations for training, 10,000 for validation and 10,000 for testing. The datasets contains *only* the positions and velocities for the "observable" particles, while the field sources are only used for visualization. Following Kipf et al. (2018), each simulation lasts for 49 timesteps. During inference, we use the first 29 steps as input and predict the remaining 20 steps.

B.2 Traffic scenes - inD

InD (Bock et al., 2020) is a real-world traffic scenes dataset that comprises trajectories of pedestrians, vehicles, and cyclists. It contains 33 recordigns, recorded at 4 different locations in Aachen, Germany. We hypothesize that discovering a latent traffic force field will be beneficial for trajectory forecasting in traffic scenes. For simplicity, we focus on static field discovery in traffic scenes. We create a subset that contains scenes from a single location. Namely, we choose "Frankenburg, Aachen", since it is the location with most interactions in the dataset. The subset corresponds to 12 recordings; we use 8 for training, 2 for validation, and 2 for testing. We follow a similar experimental setting with Graber & Schwing (2020); Kofinas et al. (2021). We divide each scene into 18-step sequences. We use the first 6 time steps as input and predict the next 12 time steps.

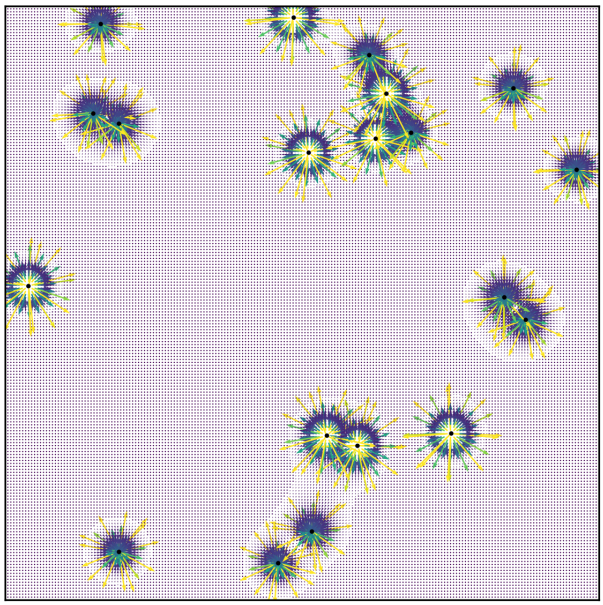


Figure 10: Static field visualization

B.3 GRAVITATIONAL N-BODY DATASET

In this experiment, we study the influence of dynamic fields, *i.e.* fields that are different across simulations. Similar to the charged particles setting, we extend the gravitational n-body dataset by Brandstetter et al. (2022) by adding gravitational sources. The equation that describes the forces is similar to eq. (58). Namely, we have

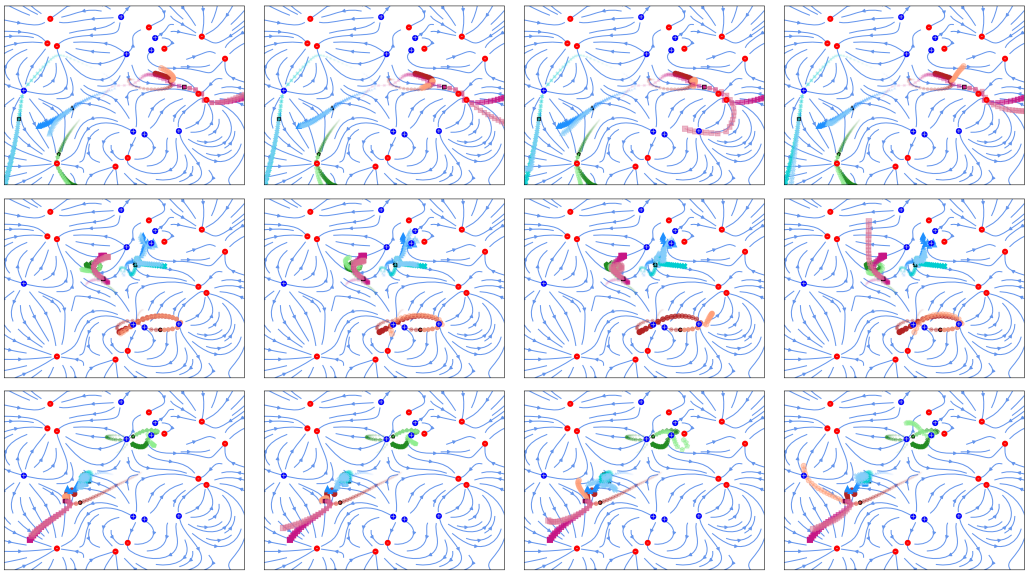
$$\mathbf{f}_{j,i}^t = C \cdot m_i m_j \frac{\hat{\mathbf{p}}_{j,i}^t}{\left\|\mathbf{p}_{j,i}^t\right\|^2}, \tag{65}$$

where m_i, m_j are the particle masses. We create a dataset of 5,000 simulations for training, 1,000 for validation and 1,000 for testing. We use $N = 5$ particles and $M = 1$ source. We set the masses of particles to $m_p = 1$, while the source has a mass of $m_s = 10$. Similarly to all our experiments, the datasets contains *only* the positions and velocities for the "observable" particles, while the field source is only used for visualization. We generate trajectories of 49 timesteps. We use the first 44 timesteps as input and predict the remaining 5 steps. All other dataset details are identical to Brandstetter et al. (2022).

C QUALITATIVE RESULTS

C.1 CHARGED PARTICLES

Figure 11 shows qualitative results on charged particles. The predictions start where markers have black edges. The markers get bigger and more opaque as trajectories evolve in time. Lighter colors indicate predictions, and darker colors indicate the groundtruth. The background streamplots indicate the groundtruth field, and are not given as input to the networks. Similarly, the blue $\oplus$ markers and the red $\ominus$ markers, are merely shown for illustrative purposes, indicating the charges of the field sources, and are not given as input to the networks.



(a) Aether (b) G-LoCS (c) LoCS (d) dNRI

Figure 11: Predictions on charged particles. Lighter colors indicate predictions, and darker colors indicate the groundtruth. Predictions start where markers have black edges. Markers get bigger and more opaque as trajectories evolve in time. The background streamplots, blue $\oplus$ markers, and red $\ominus$ markers indicate the groundtruth field, and are merely shown for illustrative purposes, they are not given as input to the networks. *Best viewed in color.*

C.1.1 DISCOVERED ELECTROSTATIC FIELD

In fig. 12 we visualize the discovered electrostatic field compared to the groundtruth one.

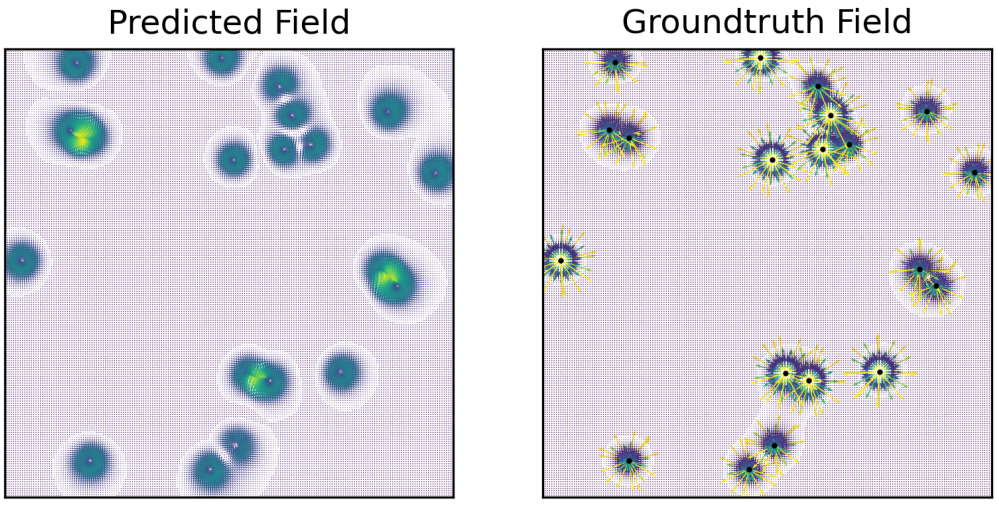
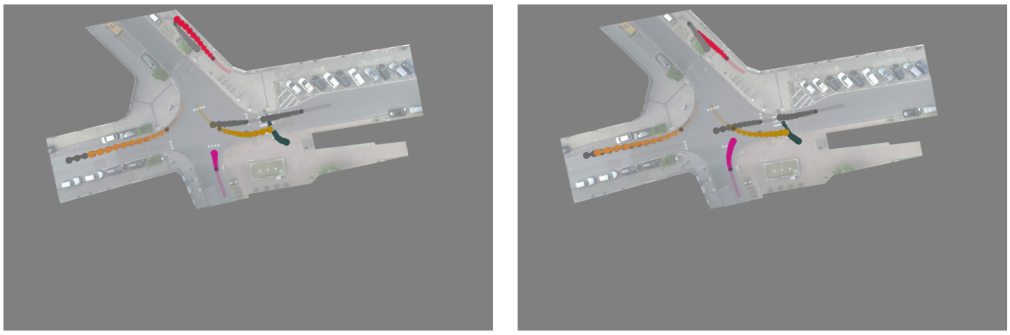


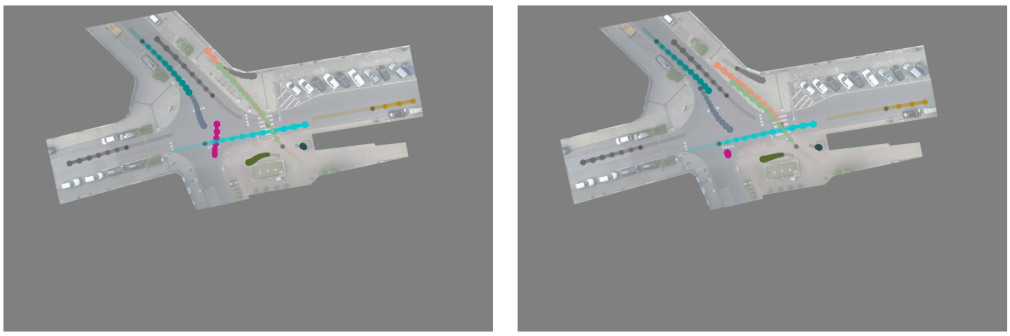
Figure 12: Learned Field (left) in charged particle settings compared to groundtruth (right).

C.2 IND

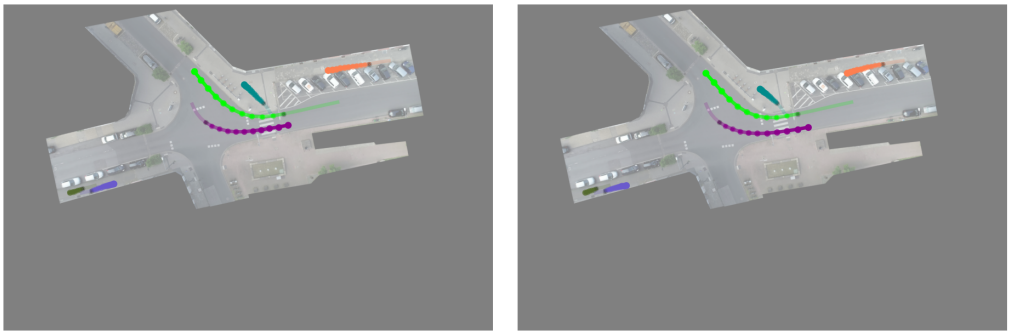
Figure 13 shows qualitative results on inD (Bock et al., 2020).



(a)

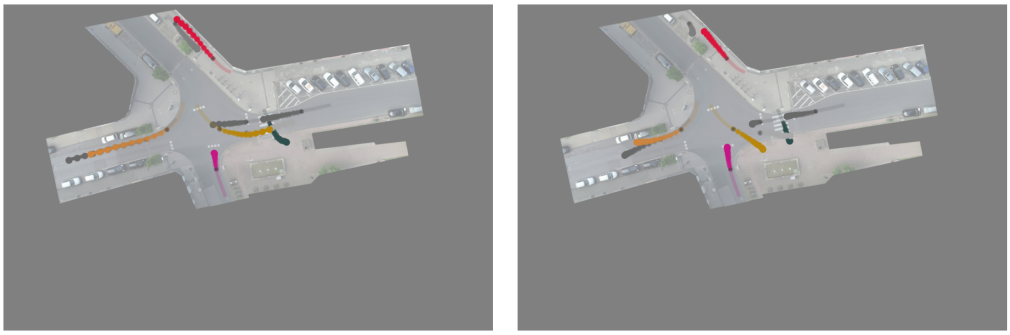


(b)

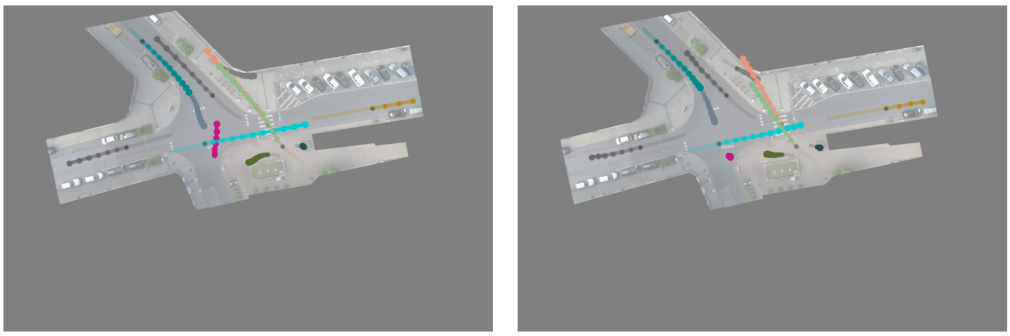


(c)

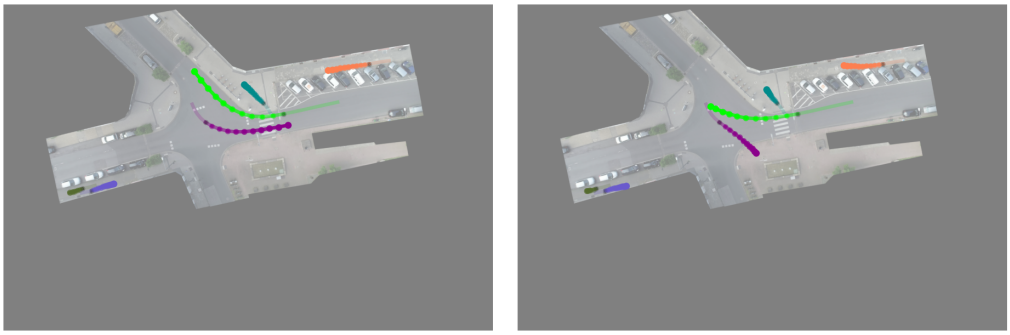
Figure 13: Aether predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*



(a)

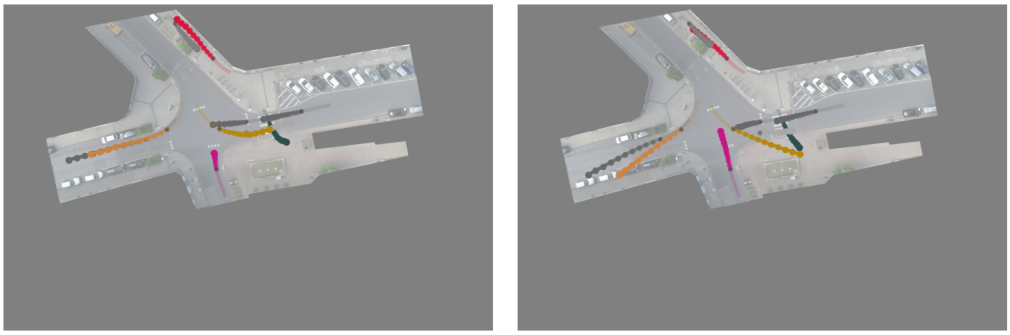


(b)

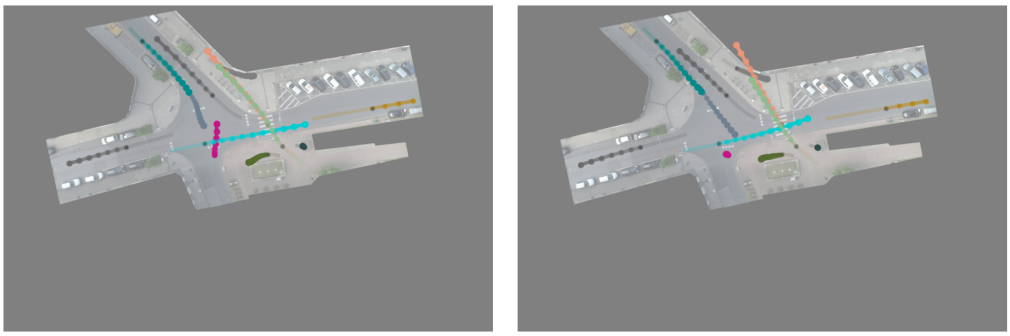


(c)

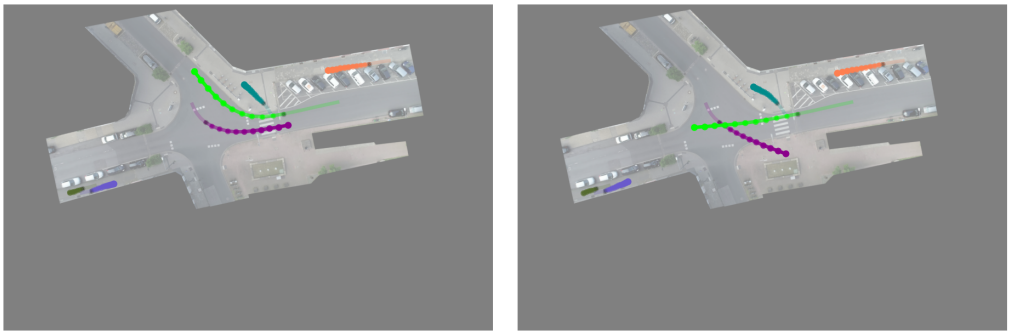
Figure 14: G-LoCS predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*



(a)

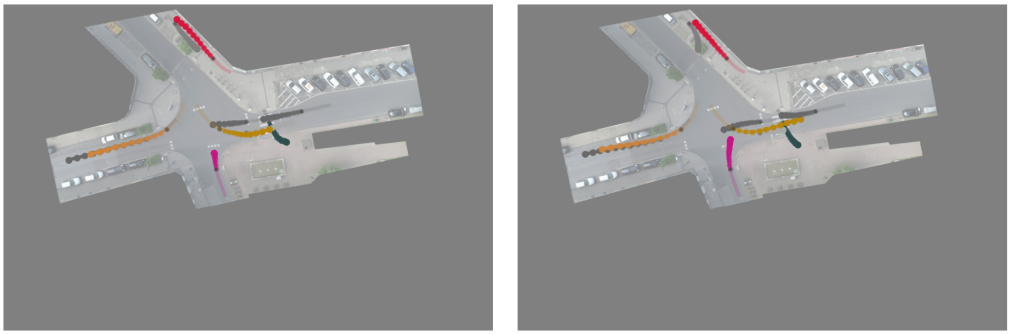


(b)

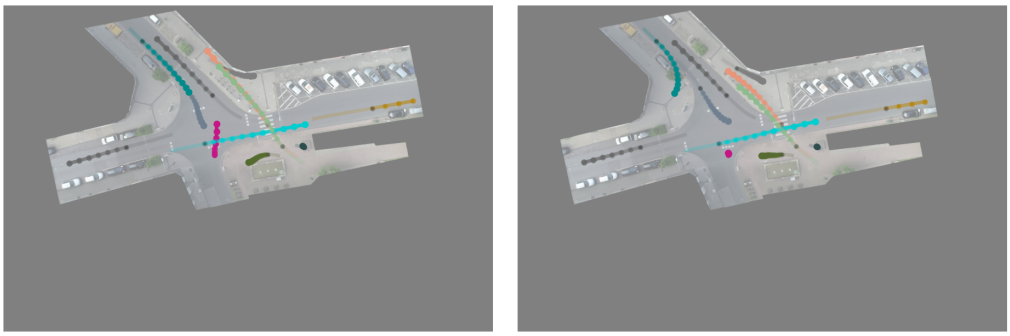


(c)

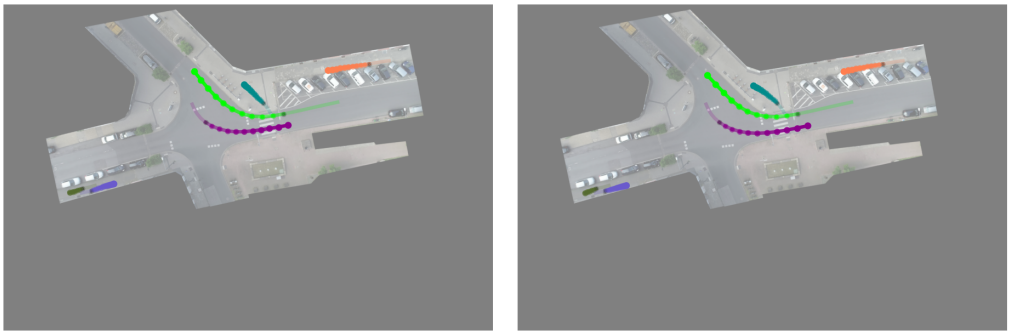
Figure 15: LoCS predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*



(a)



(b)



(c)

Figure 16: dNRI predictions (right) on inD, compared to groundtruth (left). Predictions start where markers are colored black. *Best viewed in color.*

C.2.1 DISCOVERED TRAFFIC FORCE FIELD

In fig. 17 we visualize the discovered traffic force field on inD.

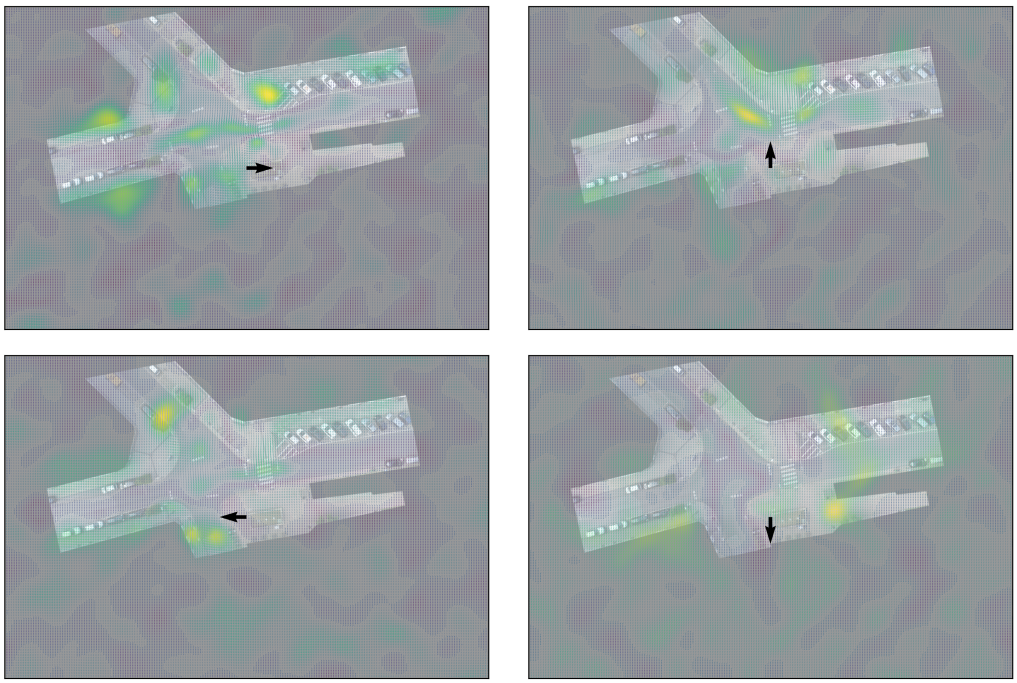
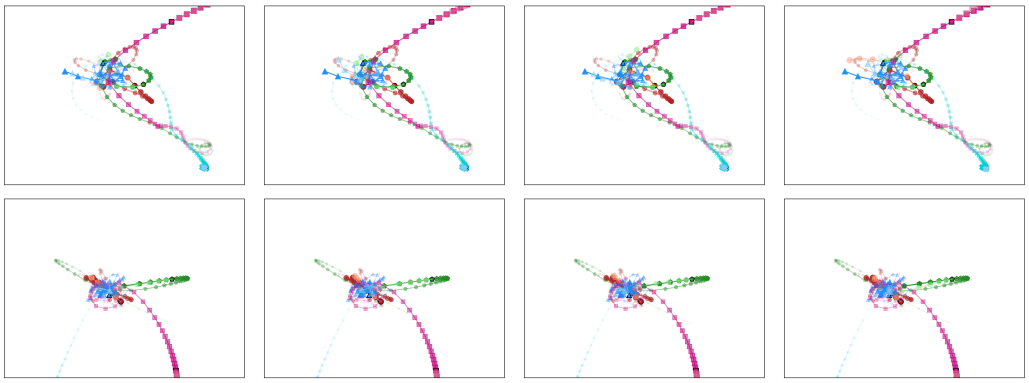


Figure 17: Discovered field on inD (Bock et al., 2020). For simplicity, we only visualize the field for discrete input orientations in $\mathrm{C}_4 = \left\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}$. *Best viewed in color.*

C.3 GRAVITY

Figure 18 shows qualitative results on the gravitational n-body problem.



(a) Aether (b) G-LoCS (c) LoCS (d) dNRI

Figure 18: Predictions on gravity. Lighter colors indicate predictions, and darker colors indicate the groundtruth. Predictions start where markers have black edges. Markers get bigger as trajectories evolve. *Best viewed in color.*

C.3.1 DISCOVERED GRAVITATIONAL FIELDS

Figure 19 shows examples of discovered fields compared to the groundtruth ones.

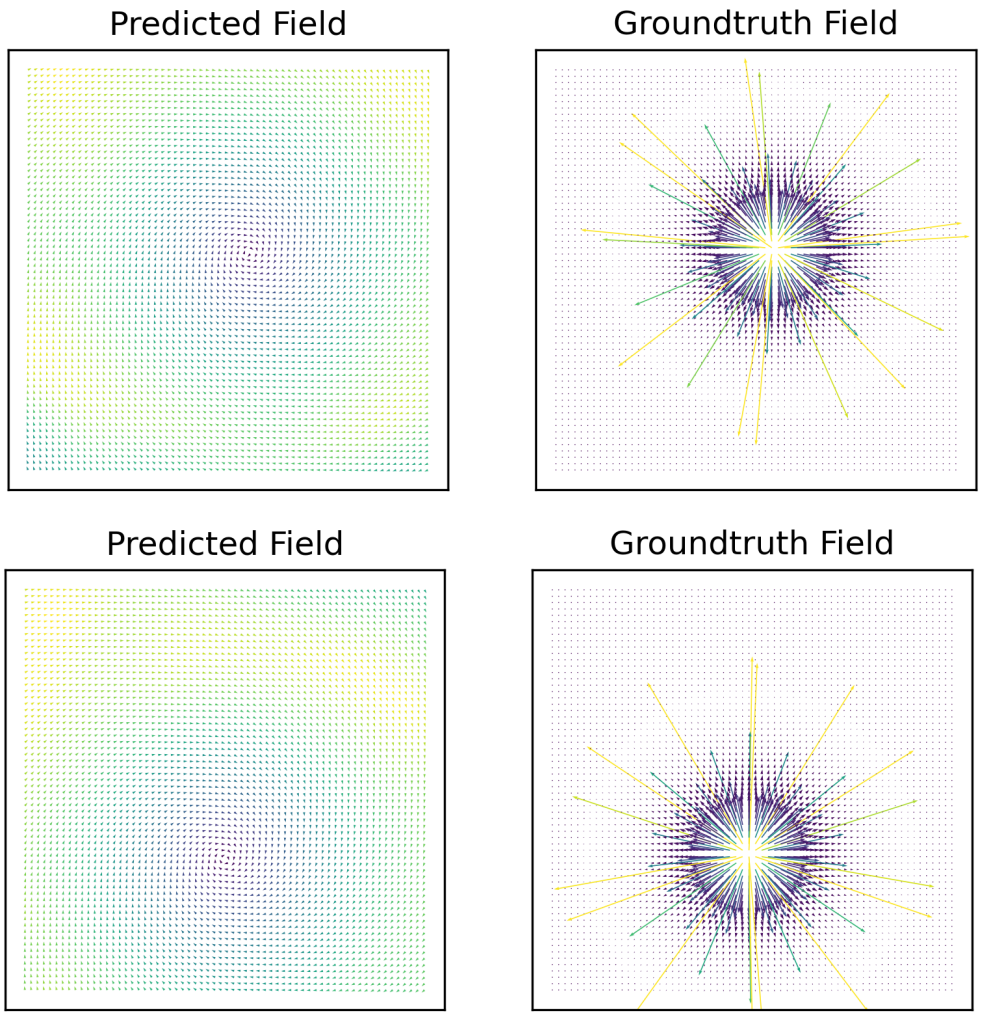


Figure 19: Learned *dynamic* fields (left) in n-body problem vs groundtruth (right).

D QUANTITATIVE RESULTS

In all settings, we report the *total errors*, *i.e.* the mean squared errors of positions and velocities over time, $E(t) = \frac{1}{ND}\sum_{n=1}^{N}\|\mathbf{x}_n^t - \hat{\mathbf{x}}_n^t\|_2^2$. Following Kofinas et al. (2021), we also separately report the L_2 norm *position errors*, $E_p(t) = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{p}_n^t - \hat{\mathbf{p}}_n^t\|_2$, and *velocity errors*, $E_u(t) = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{u}_n^t - \hat{\mathbf{u}}_n^t\|_2$.

D.1 CHARGED PARTICLES

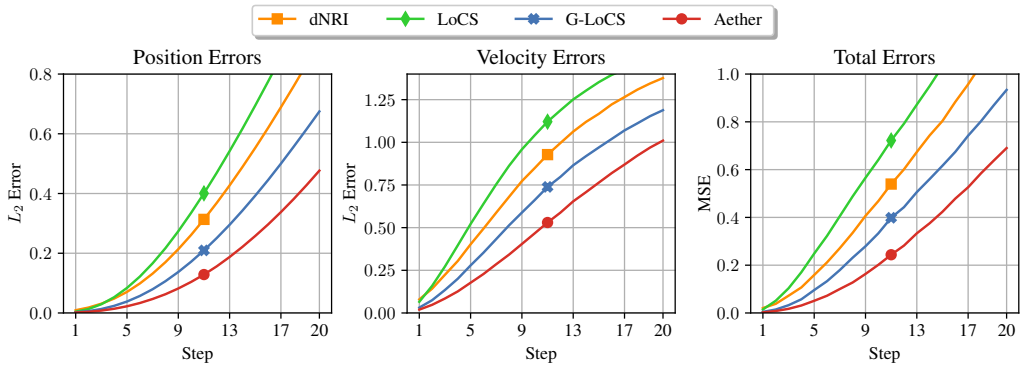


Figure 20: Results in charged particles

D.2 IND

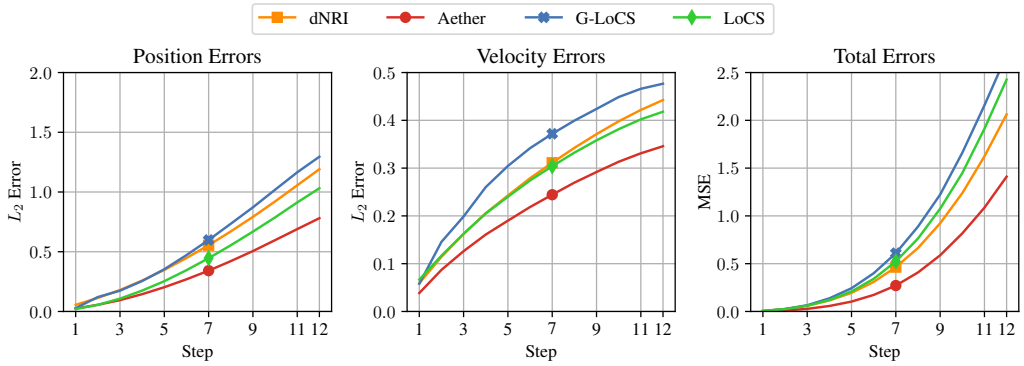


Figure 21: inD results

D.3 GRAVITY

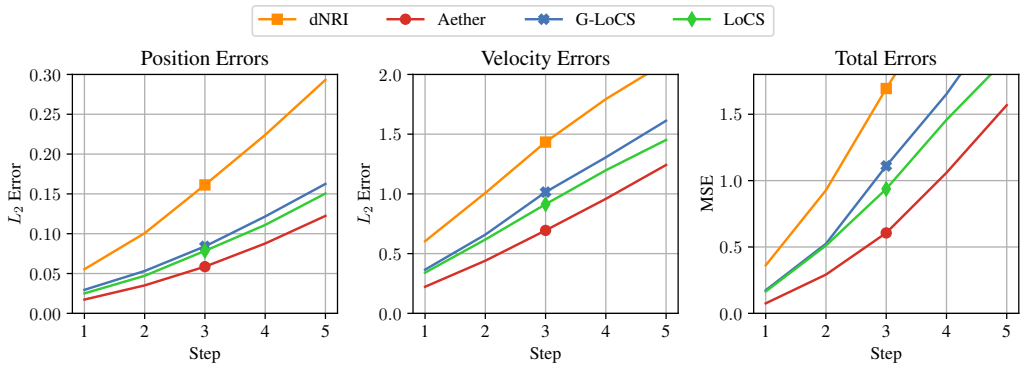


Figure 22: Gravity results

D.4 ABLATION STUDIES

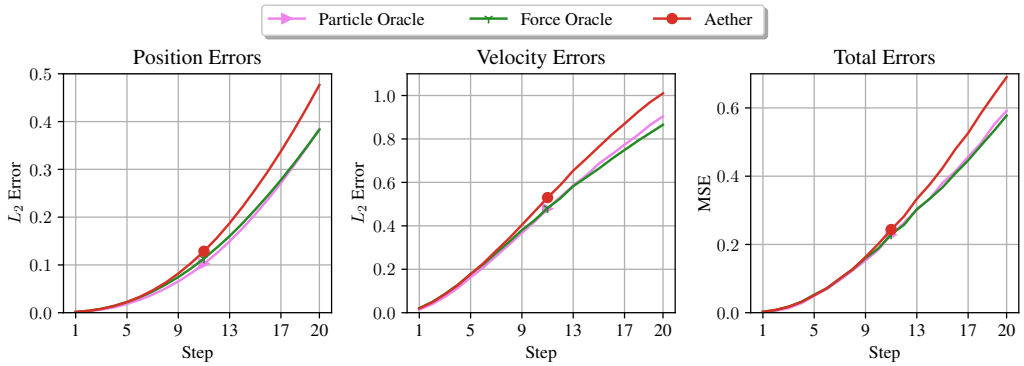


Figure 23: Ablation results in charged particles