ABSTRACT

It is challenging to predict a group of individuals’ spatiotemporal trajectories in continuous time and space, due to various environmental and intrinsic factors. Especially, social dynamics such as driving or crowding behaviors could be hard to predict due to heterogeneous and complex mapping from high-dimensional inputs to an output driven by the decision-making processes of other agents. To tackle this challenge, neural ordinary differential equations (neural ODEs) have been developed to predict continuous-time long-term dynamics with constant memory cost and high computational efficiency. Furthermore, scientific communities have developed a rich set of physics models to describe how individuals interactively make decisions. With a rapidly growing trend of employing physics-informed deep learning (PIDL) for dynamical systems in science and engineering, its application to social dynamics is understudied. This paper aims to develop an integrated framework, named “PI-NeuGODE,” that encodes physics models, complemented by symbolic regression, into neural ODEs. In the proposed model, physics informs the training of neural ODEs, while neural ODEs guide knowledge discovery. Symbolic regression is used to uncover physics knowledge from complex data. We further use graph neural networks to learn the topological interaction of individuals. The proposed method is tested on two applications, human driving and collective platooning, as well as crowding, which demonstrate the algorithmic accuracy and efficiency against baselines including existing social deep learning models.

KEYWORDS

Neural Ordinary Differential Equations; Physics-Informed Deep Learning; Graph Neural Network; Symbolic Regression

ACM Reference Format:


This work is licensed under a Creative Commons Attribution International 4.0 License.

a “social force”, which is a combination of external and internal forces. These forces include factors like social interactions, repulsion from obstacles, and attraction towards a destination. Apart from the social force assumption, another widely used assumption is that pedestrians make decisions mainly by following neighboring velocities and avoiding collision, such as the velocity obstacle model [41].

In contrast to the physics-based method, the data-driven method does not assume any prior knowledge of the underlying process governing the pedestrian dynamics. Instead, it utilizes machine learning methods to uncover the underlying patterns directly from the observed data. For example, graph neural networks, such as message passing networks [5], are widely used to capture spatial interactions. For temporal dynamics, Long Short-Term Memory (LSTM) [1] networks and attention mechanisms [14, 42] are broadly employed. Notably, neural ordinary differential equations (Neural ODE) [6] has demonstrated its proficiency in capturing dynamic system patterns, particularly in its ability to extrapolate and address the long-term prediction challenge.

Both the physics-based and data-driven methods have their pros and cons. The former is data-efficient and interpretable but may struggle with generalizing to unobserved data. The latter is generalizable but may be incapable of offering deductive insights. Also, its performance relies on huge amounts of training samples, which is sometimes inaccessible for real-world data. The fusion of these two method categories, known as physics-informed deep learning (PIDL) [11, 32], combines the merits of both the physics-based and data-driven methods and supplements their respective limitations. PIDL has been applied to multiple applications, such as car-following modeling [24, 27], crowd simulation [44], and traffic state estimation [17, 25, 35–37]. While promising, the performance of PIDL requires the assumption of the underlying physics, which is usually not available in the real-world scenario. The problem of unknown underlying physics prior knowledge is worse for multi-agent systems [13]. A potential remedy is symbolic regression, which is capable of inferring the mathematical equations that govern the dynamics. Symbolic regression has demonstrated its efficacy across diverse domains, such as materials science [3, 4] and astronomy [8].

This paper proposes, physics-informed graph neural ordinary differential equations (PI-NeuODE), for simultaneous prediction of multi-agent spatiotemporal trajectories accounting for the topological interaction over a long-term horizon. Under this framework, neural ODEs are trained to make long-term prediction of individual trajectories interacting over graphs, of which the derivatives are substituted by physics-informed deep neural networks (DNN) complemented by symbolic regression.

2 BACKGROUND AND RELATED WORK

2.1 Long-term prediction using neural ODEs

Neural ordinary differential equations (neural ODEs) are developed to predict continuous-time long-term dynamics, with constant memory cost and high computational efficiency [6]. Since its inception, neural ODEs have become a powerful tool to model complex dynamical systems. The underlying idea is to parameterize the derivative of a system’s state with a neural network (NN).

A neural ODE consists of an ordinary differential equation (ODE) of the form:

$$\frac{dx}{dt} = f_\theta(x(t), t),$$

(1)

where $x(t)$ is the system state at time $t$, $\frac{dx}{dt}$ is the time derivative, $f_\theta(x(t), t, \theta)$ is parameterized by an NN with parameters $\theta$. The loss function to train the NN is defined as:

$$L(x(t)) = L(x(t_0)) + \int_{t_0}^{t} f_\theta(x(s), t, \theta)ds$$

(2)

A neural numerical integration method, such as the Euler method or the Runge-Kutta method, is used to solve the differential equation and produce a prediction.

2.1.1 Message passing neural networks (MPNNs). MPNNs are a class of graph neural networks (GNNs) that leverage the principles of message passing algorithms to perform graph-based tasks. In MPNNs, each node passes messages to its neighbours, and these messages are used to update the node representations and aggregate information from the surrounding nodes.

Mathematically, MPNNs are modeled as functions that operate on a graph $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges. Each node is initialized with states $x_i(t_0)$, where $i \in \{1, \ldots, N\}$ and $N$ is the total number of agents. At iteration $t$, the node state $x_i(t+1)$ is updated as follows:

$$x_i(t+1) = f_u(x_i(t), \sum_{j \in \{i, \ldots, N\}} m_{ij}(t)), \quad (3)$$

where $f_u(\cdot)$ is the node update function, and $m_{ij}(t)$ is the message passed from node $j$ to node $i$ at iteration $t$. The messages $m_{ij}(t)$ are computed as follows:

$$m_{ij}(t) = f_m(x_i(t), x_j(t), e_{ij}), \quad (4)$$

where $e_{ij}$ is the edge information between nodes $i$ and $j$ and $f_m(\cdot)$ is the message passing function. The process continues until a stopping criterion is met, such as a maximum number of iterations or convergence.

2.1.2 Graph neural ODE (GODE). GODE leverages the strengths of graph neural networks (GNNs) and neural ODEs to model the complex interactions in dynamic systems. MPNN captures the spatial correlation between nodes, while the neural ODE captures the temporal evolution of the system. These two components are coupled by replacing $f$ in Eq. 1 with the message passing process defined in Eq. 4:

$$\dot{x}_i(t) = f_0(x_i(t), f_m(x_i(t), x_j(t), e_{ij}(t)), t, \theta). \quad (5)$$

Further details on the implementation of this approach will be discussed in Sec. 3.

2.2 Physics-informed deep learning (PIDL)

PIDL [32] leverages the pros of both physics-based and data-driven approaches while compensating for the cons of each. Physics-based approach refers to scientific hypotheses of what underlying physics governs observations, like the first principle, which is data-efficient and interpretable but may not well capture complex data patterns.
In contrast, the data-driven approach does not bear any prior knowledge of how things work and how different quantities are correlated. Instead, they rely on machine learning techniques such as deep neural networks (DNN) to learn and infer patterns from huge amounts of training samples, but require inductive bias for unseen data. Recent years have seen a rapidly growing trend of applying PIDL to dynamical systems in science and engineering, for its power in robust prediction [2, 9, 20].

To incorporate PIDL into neural ODE, the derivative of a dynamical system contains both the known (represented by physics) and unknown information (represented by PUNN). [28, 38] further incorporates stochasticity into the prediction. These studies, however, are not focused on social dynamics but more on physical processes.

2.3 Symbolic regression
Symbolic regression aims to discover mathematical expressions to match a given dataset [34]. When combined with DNNs informed by physics, symbolic regression demonstrates its robustness in discovering part of physics equations [30, 33, 40, 44].

The mathematical functions are usually represented as a general expression with variables and operators. The task of symbolic regression is to find the optimal values of the coefficients that result in the best fit of the function to the observed data. To perform symbolic regression, some optimization algorithm is applied to the function and the coefficients are iteratively adjusted until the best fit is obtained. Metrics, such as mean squared error or correlation coefficient, are used to determine the goodness of fit of the function to the observed data.

2.4 Related work and contributions
In a nutshell, there are studies that have integrated PIDL into symbolic regression [30, 33, 40, 44, 44], PIDL into neural ODEs [28, 38], or GNN into neural ODEs [7, 18, 31, 45]. However, none has bridged all three methodologies into one unified framework, which is the focus of this paper. The closely related studies are [18, 43, 44, 44] combined two neural ODEs, one for temporal processing and the other for spatial processing, for spatiotemporal traffic forecasting. However, the existing domain knowledge that uses ODEs/PDEs for traffic evolution is not accounted for. [44] applied PIDL, symbolic regression, and GNN to crowd modeling, and a student-teacher co-training algorithm was developed for multiple-step rollout, which helps generalize the original single-step prediction. The same issue exists in [43], where the one-step prediction is repeatedly conducted to achieve the long-term prediction. In contrast, our paper employs neural ODE for multiple-step prediction, which is more computationally efficient and scalable.

Our main contributions are: (1) we develop a unified framework of integrating PIDL and symbolic regression into graph neural ODEs, which can capture multi-agent, long-term prediction of social dynamics. (2) we develop an algorithm to train the data-driven component (i.e., GODE) and the physics-informed components (i.e., PIDL) simultaneously, rather than calibrate the physics models prior to the model training, and (3) the efficiency of the proposed algorithms is demonstrated on two types of dynamics, platooning and crowding, using both hypothetical and real-world datasets.

The rest of this paper is organized as follows: Sec. 3 introduces the integrated methodology framework and presents the algorithm. Secs. 4-5 demonstrate the performance of our algorithms on two scenarios, human driving and platooning (via a neighbouring interaction), and crowding (via a graph interaction). Sec. 6 concludes and points out future directions.

3 METHODOLOGY
3.1 Problem statement
Denote the system state as $X(t) = \{x_i(t)\}_{i=1}^{N}$, which contains the states of all agents. The movement of an agent at each time step $t$ is assumed to be governed by some underlying control signal. Thus, the spatiotemporal trajectory is the result of the consecutive control signals or decision-making processes. Consider the multi-agent scenario, this process can be depicted as:

$$X(t) = X(t_0) + \int_{t_0}^{t} f(X(t), t, \theta) dt,$$

where $f(\cdot)$ is the underlying decision-making model that is inaccessible. The trajectory prediction problem is to find a surrogate decision-making model $f_\theta$ such that the predicted state error is minimized:

$$\min_{\theta} \int_{t_0}^{t} (X(t) - \hat{X}(t))^2$$

s.t. $\hat{X}(t) = X(t_0) + \int_{t_0}^{t} f_\theta(X(t), t, \theta) dt.$

We focus on two problems of the spatiotemporal trajectory predictions, i.e., the initial value problem (IVP) and the sequence-to-sequence (seq2seq) learning problem. Each problem is detailed below:

(1) IVP involves making predictions based on only the initial states of the system, represented by $X(t_0)$, and the boundary conditions. This approach can be viewed as a trajectory prediction problem with limited data, as it only takes into account the initial states but not the historical information of the system.

(2) Seq2seq prediction utilizes historical observations of the system to make predictions about future states. In this approach, the observed trajectory is divided into segments, with the goal of using the previous segments to predict the next ones.

The IVP and seq2seq problems will be demonstrated using the platoon modeling and pedestrian trajectory prediction experiments, respectively. Both numerical and real-world datasets will be used in both experiments.

3.2 Model architecture
The proposed method, as depicted in Fig. 2, consists of two components. The first component is the data-driven component, which incorporates a neural ODE model, encoded with a message-passing type graph neural network. The second component is the physics-informed component, which includes a partially learned physics equation obtained through symbolic regression. The integration of these two components allows for a well-balanced solution between data-driven and physics-based approaches.

3.2.1 Graph neural ODEs (GODE). The GODE structure is illustrated in Fig. 2. To simplify the notation, we utilize the symbols
1, . . . , t to represent the time index \( t_0, . . . , t \). The GODE model consists of three main components: an encoder, a neural ODE model, and a decoder. In this framework, the encoder replaces the message passing function in Eq. 4 by transforming the historical states segment \( x(1 : t) \) into a latent variable \( z_\theta(t) \) of the ego-agent, while taking into account the states of its neighbours. This latent variable is then used as the initial condition for the neural ODE model, which calculates the future latent solution \( z_\theta(t + 1 : t + T) \). Finally, the decoder converts the latent solution back to the data space. The encoder and decoder can be implemented as either multi-layer perceptrons or recurrent networks like GRUs. The implementation details of these components for IVP and seq2seq problems are described in Secs. 4 and 5, respectively. The estimated trajectory \( \hat{X}(t + 1 : t + T) \) is compared to the ground truth, and the observation loss is calculated accordingly:

\[
L_o(\theta) = \frac{1}{T} \sum_{t=t+1}^{T+t} ((\hat{X}(t)) - X(t))^2 dt. \tag{8}
\]

### 3.2.2 Physics-informed deep learning

**Definition 3.1. Physics-informed deep learning (PIDL).**

Denote the (labeled) observation \( O \) and the (unlabeled) collocation points \( C \) below:

\[
\begin{cases}
O = \{ X(t); t \in T_o \} : \text{within-domain observation}, \\
C = \{ \hat{X}(t) = \text{ODESolver}(X(t_0), f_\lambda, t_0, t, \lambda); t \in T_c \} : \text{collocation points},
\end{cases}
\]

where, \( T_o \) is a set of time steps at which the state \( X(t) \) is observed; \( T_c \) is a set of time steps at which a physics-informed computational graph (PICG), denoted as \( f_\lambda(X(t), t|\lambda) \), for computing the time derive of \( \hat{X}(t) \), denoted as \( \hat{X}(t) \).

In summary, a PIDL model, denoted as \( f_\theta,\lambda(X(t), t|\theta, \lambda) \), is commonly represented by a hybrid of two graphs, namely, the PUNN and the PICG. It aims to train an optimal parameter set \( \theta^* \) for PUNN and an optimal parameter set \( \lambda^* \) for PICG. For simplicity, we let observation and collocation time steps be the same, i.e. \( T_o = T_c = \{ t + 1, . . . , t + T \} \).

In our framework, the PUNN is the GODE model introduced above. The PICG is spanned by physics knowledge, in which nodes are mathematical quantities, edges are operators connecting two quantities, and a path represents a relation from a starting quantity to a target one [27]. PICG could be represented by a graph with known physics equations, or partial knowledge complemented by additional mathematical operators to be learned via symbolic regression [44].

In summary, the PUNN is replaced by GODE that represents a sequential decision-making process on graphs. The PICG is expanded with symbolic regression to learn unknown physics knowledge that could dominate the partial information from observational data. PICG can be generally represented by \( f_\lambda(x) = f_{\text{phy}}(x) + w^T x \), where \( f_{\text{phy}}(\cdot) \) is the known physics from prior knowledge; \( x \) is a vector of input variables and \( w \) is a vector of coefficients together with numerical operators such as \( +, - , \times, \), and \( \div \) to be learned by symbolic regression. Mathematically, the PICG takes collocation points as input and produces augmented solutions, i.e., \( X(t) = X(t_0) + \int_{t_0}^{t} f_\lambda(X(t), t, \lambda)dt \). These collocation points are also fed into the PUNN for the solution \( \hat{X}(t) = X(t_0) + \int_{t_0}^{t} f_\theta(X(t), t, \theta)dt \).

Based on these two solutions, the physics loss is defined as:

\[
L_\epsilon(\theta, \lambda) = \frac{1}{T} \sum_{t=t+1}^{T+t} ((\hat{X}(t)) - X(t))^2. \tag{10}
\]
### 3.3 Training Algorithm

The final loss function is defined as
\[
\text{Loss}_{\theta, \lambda} = \alpha \cdot L_o(\theta) + \beta \cdot L_c(\theta, \lambda),
\]
where \(\alpha, \beta\) are weights of the observation loss and the physics loss, respectively. The detailed training algorithm is shown in Alg. 1.

**Algorithm 1 Training Algorithm for PI-NeuGODE.**

**Required:** Training iterations Iter; Learning rate lr; Loss function weights \(\alpha, \beta\).

**Input:** The observation data \((X(t), t \in T_o)\) and collocation points \((X(t), t \in T_c)\).

1. for \(k \in \{0, ..., \text{Iter}\} \) do
2. Sample the sequence states \(X(1 : t, t + 1 : t + T)\), from the batch.
   // generate PUNN solutions
3. \(z(t) = \text{Encoder}(X(t))\)
4. \(z(t + T) = z(t) + \int_{t}^{t+T} f_0(z(t), t, \theta) dt\)
5. \(\hat{X}(t + T) = \text{Decoder}(z(t + T))\)
6. Generate \(\tilde{X}(t + 1 : T)\) similarly using the collocation data
   // generate PICG solutions
7. \(\tilde{X}(t + T) = X(t) + \int_{t}^{t+T} f_1(X(t), t, \lambda) dt\)
   // update the PUNN
8. Calculate \(L_o\) by Eq. 8, \(L_c\) by Eq. 10
9. \(\theta^{k+1} \leftarrow \theta^k - lr \cdot \text{Adam}(\theta^k, \nabla_{\theta}(\alpha L_o + \beta L_c))\)
   // update the PICG
10. Conduct symbolic regression by minimizing \(L_c\)
end for

In the subsequent sections, we present a comprehensive evaluation of the proposed model through two experiments. Firstly, we assess its performance in solving the IVP problem in the context of vehicle platoon modeling, utilizing both numerical and real-world data. The numerical experiment is designed to verify the ability of the proposed model to accurately identify the ground-truth equation in a controlled environment. This is followed by a validation experiment, which uses real-world data to demonstrate the model’s efficacy in capturing real-world dynamics. Finally, we demonstrate the proposed model’s ability to solve the seq2seq problem by applying it to the pedestrian trajectory prediction problem, using both numerical and real-world data. We will also present the specific architectures employed for each problem.

### 4 EXPERIMENTS: PLATOON MODELING

We first briefly introduce the problem formulation as follows: In a Platoon Modeling (PM) problem, a platoon of vehicles forms a multi-agent system, where each vehicle decides to accelerate, brake or cruise depending on its relation with its leader. Denote the state of the \(i\)th vehicle at time step \(t\) as \(x_i(t) = [p_i(t), v_i(t)]\), where \(p_i(t)\) is the longitudinal position and \(v_i(t)\) is the velocity. The dynamic of the followers can be depicted as:

\[
\begin{align*}
\dot{p}_i(t) &= \frac{dp_i(t)}{dt} = v_i(t) \\
\dot{v}_i(t) &= \frac{dv_i(t)}{dt} = f_{\lambda i}(x_i, x_j), \quad i \in \{1, ..., N\}.
\end{align*}
\]

where \(f_{\lambda i}\) is the control strategy of the \(i\)th agent given the states of its leader and itself. Given the trajectory of the first leading vehicle \(x_0(1 : t + T)\), the task of the PM is to predict the trajectory of all its followers. Thus, the state of the PM system at time \(t\) is \(X(t) = \{x_i(t)\}_{i=1}^{N} \in \mathbb{R}^{N \times 2}\).

The PM is an example of IVP, where only the initial state \(X(1)\) and the platoon leader’s trajectory \(x_0(1 : t + T)\) are known. A platoon can be represented as a chain-like graph, where each vehicle only has one neighbouring vehicle, which is its immediate leader.

#### 4.1 GODE Structure

Fig. 3 illustrates the GODE structure that is modified for the PM. The initial states of a vehicle and its leader are used as input to update the state of the follower. This GODE framework for IVP does not require an encoder or decoder as only one-time-step state information is used. The direct output of the network \(f_{\lambda i}\) already contains meaningful information, as it represents the derivatives of the vehicle states. In the neural ODESolver, a neural network \(f_{\lambda i}\) is trained to learn the mapping from the features of two consecutive vehicles to estimated acceleration \(\dot{a}\) of the follower. The estimated acceleration \(\dot{a}\) is used to update the state of the platoon by the Euler method.

#### 4.2 Dataset

**Numerical Dataset:** In our numerical experiment, the platoon data is generated by the intelligent driving model (IDM) [39], a well-known model in the transportation domain. The IDM models the longitudinal motion of vehicles on a highway, considering their acceleration and deceleration behavior based on their speed and the gap to the leading vehicle. We assume each vehicle in the platoon follows the IDM equation. The complete form of the IDM equation...
is shown below:

\[
\begin{align*}
X_i(t) &= v_i(t) \\
\dot{v}_i(t) &= a_{\text{max}} \left[ 1 - \frac{v_i(t)}{a_{\text{max}}} \right] \left( \frac{\Delta v_i(t)}{\Delta t} \right)^2
\end{align*}
\]

where \(X_i(t)\) is the position of the \(i\)th vehicle at time step \(t\), \(v_i(t)\) is the velocity, \(\Delta v_i = x_i - x_{i-1}\) is the gap to the leading vehicle, and \(\Delta t\) is the time difference. This equation has 5 parameters: \(v_{\text{des}}, T_0, s_0, a_{\text{max}}, b\) where \(v_{\text{des}}\) is the desired velocity, \(T_0\) is the desired time headway, \(s_0\) is the minimum spacing in congested traffic, \(a_{\text{max}}\) is the maximum acceleration allowed, and \(b\) is the comfortable deceleration, respectively. All vehicles are assumed to share the same set of parameters, i.e., \([v_{\text{des}}, T_0, s_0, a_{\text{max}}, b] = [30 \text{ m/s}, 0.75 \text{ s}, 2.15 \text{ m}, 2 \text{ m/s}^2, 4 \text{ m/s}^2]\). More numerical dataset details are included in the supplementary material.

**Real-world Dataset:** The real-world data is from the Next Generation SIMulation (NGSIM) dataset\(^1\), which is an open dataset that collects vehicle trajectories every 0.1 seconds. We focus on the US Highway 101. More dataset details are shown in the supplementary material.

### 4.3 Setting

**Baselines:** In this study, we compare the performance of our proposed model with several baseline models including SocialGAN \([15]\) and SocialLSTM \([1]\). The SocialGAN model is a generative adversarial network (GAN) based approach for modeling the interactions between pedestrians in a crowd. The SocialLSTM model is a long short-term memory (LSTM) network that is trained to predict the future trajectories of individuals in a crowd by incorporating the interactions between individuals. Furthermore, we also consider the simple baseline of independent decision-making, which is IDM, where each vehicle acts as an independent agent following its own trajectory computed by Eq. 13.

**Evaluation Metrics:** In this study, we evaluated the performance of our proposed model using mean squared error (MSE) and mean absolute error (MAE). Detailed settings are included in the supplementary material.

### 4.4 Results

**Performance Comparison:** In Tab. 1, we make a comparison of four models: PI-NeuGODE, SocialGAN, SocialLSTM, and IDM by looking into MSE and MAE. We leverage noise-free and noisy data to evaluate all models. It is shown that our proposed PI-NeuGODE outperforms other models on both data sets. The results show that among all four models, PI-NeuGODE has the lowest MSE and MAE (marked in bold), while IDM has the highest MSE and MAE for the NGSIM data.

**Effect of Varying Training Size:** To demonstrate the generalizability of our model with limited labeled data, we visualize the performance of PI-NeuGODE and SocialGAN when varying training size of the car-following dataset in Fig. 4. We can see that the results of PI-NeuGODE do not deteriorate as drastically as those of SocialGAN as we reduce the training size. This is because PI-NeuGODE uses both labeled and unlabeled instances in training.

**Table 1: Evaluation of different models for the platoon modeling problem**

<table>
<thead>
<tr>
<th>Model</th>
<th>Noise-free MSE</th>
<th>Noise-free MAE</th>
<th>Noisy MSE</th>
<th>Noisy MAE</th>
<th>NGSIM MSE</th>
<th>NGSIM MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SocialGAN</td>
<td>1.45</td>
<td>1.27</td>
<td>2.45</td>
<td>1.60</td>
<td>3.38</td>
<td>2.12</td>
</tr>
<tr>
<td>SocialLSTM</td>
<td>2.56</td>
<td>1.78</td>
<td>4.56</td>
<td>2.39</td>
<td>3.51</td>
<td>2.69</td>
</tr>
<tr>
<td>IDM</td>
<td>1.28</td>
<td>1.31</td>
<td>1.89</td>
<td>1.50</td>
<td>4.64</td>
<td>2.93</td>
</tr>
<tr>
<td>PI-NeuGODE</td>
<td>1.23</td>
<td>1.16</td>
<td>1.34</td>
<td>1.18</td>
<td>2.62</td>
<td>1.46</td>
</tr>
</tbody>
</table>

**Figure 4: Effect of the training size.**

leading to the learning of more generalizable solutions compared to socialGAN that only uses labeled instances.

**Recovered Physics:** For the numerical data, the discovered physics is of the same form of Eq. 13, and the estimated coefficients are: \([v_{\text{des}}, T_0, s_0, a_{\text{max}}, b] = [28.45 \text{ m/s}, 0.70 \text{ s}, 4.25 \text{ m}, 2.14 \text{ m/s}^2, 3.24 \text{ m/s}^2]\).

For the symbolic regression of the NGSIM platooning trajectory, the operator set is \(\{+, -, *, /, \sqrt(), (\cdot)^2, (\cdot)^3, (\cdot)^4\}\). To control the complexity of the learned equation, we add a constraint that the power operator, i.e., \((\cdot)^2, (\cdot)^3, (\cdot)^4\), can only be used once and cannot be used with each other. The discovered equation is:

\[
\dot{v}_i(t) = 0.063(0.042v_i(t) - 0.072u_i(t))^2 - 0.041u_i(t)^2 - 0.083\dot{v}_i(t) + 0.14,
\]

where the variables share the same meaning as in Eq. 13. For comparison, we apply the symbolic regression to the NGSIM data directly, and the optimal learned equation is a constant \(\dot{v}_i(t) = 0.025\). This is because the real-world data is noisy, and exploring higher-order operators tends to cause high empirical risk. From this result, we can see that the neural network serves as a filter to smooth the real-world data, allowing subtle patterns to be learned through symbolic regression.

**Visualization:** Fig. 5a and 5b plot the trajectories for a platoon of vehicles predicted by SocialGAN and PI-NeuGODE, respectively. The red line is the trajectory of the leading car. The solid and dashed blue lines represent the actual and predicted trajectories of the following cars, respectively. It is shown that PI-NeuGODE outperforms SocialGAN when predicting trajectories for the next 5 seconds. The gap between solid blue lines keeps increasing, as the predicted time progresses, as seen in Fig. 5a. This trend is invisible in Fig. 5b. Thus, compared to SocialGAN, PI-NeuGODE has a higher
5 EXPERIMENT: PEDESTRIAN TRAJECTORY PREDICTION

In this section, we further apply PI-NeuGODE to the pedestrian trajectory prediction problem, which is an example of the seq2seq problem and is important for a wide range of real-world applications such as crowd management, autonomous navigation, and video surveillance.

5.1 GODE Structure

Different from IVP which only uses the initial state, seq2seq prediction observes a sequence of states, and predicts the future states sequentially. Solving the pedestrian trajectory prediction problem as a seq2seq prediction problem, the structure of neural ODESolver is illustrated in Fig. 6.

5.2 Dataset

Numerical Dataset: The trajectory of the pedestrian is governed by the social force model (SFM) [16] below

\[
\frac{d^2 p_i}{dt^2} = F_{ID} + \sum_{j \in \{1, \ldots, N\}} \sum_{(i, j) \in E} F_{ij}^E + \sum_{j \in O} F_{ij}^O, \tag{15}
\]

where \(E\) and \(O\) are the sets of edges and obstacles, respectively; \(F_{ID}\), \(F_{ij}^E\), and \(F_{ij}^O\) represents the traction force of destination \(D\), the repulsive force of pedestrian and obstacle \(j\) on pedestrian \(i\), respectively.

Those three forces can be further depicted as:

\[
F_{ID} = m_i \frac{\mathbf{v}_i \cdot \mathbf{n}_{ID}}{\tau} - \mathbf{v}_i(t)
\]

\[
F_{ij} = \lambda_1 e^{-\frac{d_{ji}(t)}{\lambda_2}} \cdot \mathbf{n}_{ji}
\]

\[
F_{oi} = \lambda_3 e^{-\frac{d_{oi}(t)}{\lambda_4}} \cdot \mathbf{n}_{oi}, \tag{16}
\]

where \(v_{id} = 0.5 m/s\) is the desired walking speed. \(n_{ID}\) is the unit vector to the target direction. \(m_i\) is the mass of pedestrian \(i\), and \(\tau = 0.4 s\) is the simulation time step. \(\lambda_1\) and \(\lambda_2\) are tunable parameters with a physical meaning of force intensity and force radius, respectively. More implementation details are shown in the supplementary material.

Real-world Dataset: We apply our method to the UCY [21] and ETH [29] datasets, which are classic benchmarks for pedestrian trajectory prediction in computer vision research.

5.3 Setting

Baselines: In addition to SocialGAN and SocialLSTM, we include the social force model (SFM) as a baseline method in our evaluation. Two attention-based methods, the AgentFormer [42] and the Transformer TF [14], are included for comparison. Moreover, we investigate the effectiveness of Graph Neural ODE (GODE) [31] and the Coupled Graph Neural ODE (CGODE) [18]. To assess the generalizability of our physics-informed framework, we apply the physics-informed technique to CGODE and introduce the physics-informed CGODE (PI-CGODE) as another baseline.

5.4 Results

Table 2: Evaluation of different models for the pedestrian trajectory prediction problem

<table>
<thead>
<tr>
<th>Model</th>
<th>Numerical Data</th>
<th>ETH (Mean)</th>
<th>UCY (Mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>SFM</td>
<td>0.24</td>
<td>0.42</td>
<td>0.84</td>
</tr>
<tr>
<td>SocialGAN</td>
<td>0.38</td>
<td>0.56</td>
<td>0.64</td>
</tr>
<tr>
<td>SocialLSTM</td>
<td>0.51</td>
<td>0.73</td>
<td>0.71</td>
</tr>
<tr>
<td>Social-STGCNN</td>
<td>0.31</td>
<td>0.54</td>
<td>0.52</td>
</tr>
<tr>
<td>AgentFormer</td>
<td>0.25</td>
<td>0.47</td>
<td>0.42</td>
</tr>
<tr>
<td>Transformer TF</td>
<td>0.28</td>
<td>0.51</td>
<td>0.44</td>
</tr>
<tr>
<td>GODE</td>
<td>0.26</td>
<td>0.45</td>
<td>0.75</td>
</tr>
<tr>
<td>CGODE</td>
<td>0.23</td>
<td>0.42</td>
<td>0.45</td>
</tr>
<tr>
<td>PI-CGODE</td>
<td>0.20</td>
<td>0.34</td>
<td>0.36</td>
</tr>
<tr>
<td>PI-NeuGODE</td>
<td>0.16</td>
<td>0.32</td>
<td>0.31</td>
</tr>
</tbody>
</table>

Performance Comparison: The experimental results for pedestrian trajectory prediction in Tab. 2 demonstrate the superior performance of our model compared to baselines. We note that the results of CGODE and PI-CGODE are not as good as our proposed model. Upon carefully analyzing the differences between CGODE and our proposed method, we believe that it is due to the different types of graph neural networks used. CGODE employs a GCN-based graph neural network with a binary adjacency matrix to represent the graph structure, which may not be sufficient to capture the
complexity of pedestrian interaction. In contrast, message-passing graph neural networks are able to model the rapid change in the relative distance and direction between two pedestrians.

**Visualization:** Fig. 7a and 7b plot the trajectories for 4 pedestrians predicted by SocialGAN and PI-NeuGODE, respectively. In each figure, 10 seconds of trajectories are used as observation to predict the next 5 seconds. The ground truth is represented by blue lines while the predictions of both models are depicted by red dashed lines. From the figures, it can be seen that PI-NeuGODE outperforms S-GAN and is closer to the ground truth.

Fig. 8 presents the predicted pedestrian trajectory for the UCY (left) and ETH (right) datasets. The error is aggregated for all scenarios in each dataset. The solid yellow lines, solid green lines, and dashed red lines represent the input trajectories, ground-truth output trajectories, and predicted output trajectories, respectively. The figures show that there is a high level of agreement between the predicted and ground-truth trajectories in both datasets.

**Computational Time:** The computational time required for the proposed model, along with the baseline models, is presented in Tab 3. SocialGAN, SocialLSTM, and GODE are selected for their computational efficiency, making them suitable benchmarks for comparison. As can be observed, PI-NeuGODE has a similar computation cost as GODE and SocialGAN. SocialLSTM achieves a faster computation compared to other models, but this fast computation is at a cost of model accuracy as indicated in Tab. 2. The results indicate that the computational effort required for our model is negligible, thus confirming that it is not a limiting factor.

### Table 3: The computation time for the trajectory prediction

<table>
<thead>
<tr>
<th>Model</th>
<th>Computation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training (s/epoch)</td>
</tr>
<tr>
<td>PI-NeuGODE</td>
<td>28.5</td>
</tr>
<tr>
<td>GODE</td>
<td>23.6</td>
</tr>
<tr>
<td>SocialGAN</td>
<td>20.2</td>
</tr>
<tr>
<td>SocialLSTM</td>
<td>12.6</td>
</tr>
</tbody>
</table>

**Ablation study:** In order to evaluate the contributions of each component in our proposed model, we conducted an ablation study as shown in Fig. 9. PI-LSTM” refers to applying PIDL to an LSTM model. “w/o PI” and “w/o symbolic regression” are the variants of our proposed PI-NeuGODE without using PIDL and symbolic regression, respectively. The experiment setting is the same as in the numerical trajectory prediction task. The results clearly show that the integration of the physics-informed technique significantly improves the performance of the model. However, when combining PIDL and GODE without incorporating symbolic regression, we observed no noticeable improvement in performance. Specifically, the outcomes indicate that the combined model achieves performance levels similar to that of the PI-LSTM, suggesting that a mere combination of these two techniques does not offer additional advantages. This finding underscores the indispensability of symbolic regression as an essential component within the model architecture.

### 6 CONCLUSION

We develop a first-of-its-kind methodological framework integrating PIDL, graph neural ODEs, and symbolic regression, for the prediction of interacting spatiotemporal trajectories. The integrated framework can not only make long-term prediction using historical data, but also leverage physics knowledge to learn individuals’ dynamic decision-making processes with symbolic regression. The effectiveness of the proposed model has been validated on two tasks, human driving and platooning together with crowding. This work can be extended by modeling stochasticity and enhancing the model generalization using transfer learning.

### ACKNOWLEDGMENTS

This work is sponsored by the National Science Foundation (NSF) under NSF SCC-2218809.