Neural ODE-based Imitation Learning (NODE-IL): Data-Efficient Imitation Learning for Long-Horizon Multi-Skill Robot Manipulation

Shiyao Zhao[1], Yucheng Xu[1], Mohammadreza Kasaei[1], Mohsen Khadem[1], Zhibin Li[2]

***Abstract*— In robotics, acquiring new skills through Imitation Learning (IL) is crucial for handling diverse complex tasks. However, model-free IL faces challenges of data inefficiency and prolonged training time, whereas model-based methods struggle to obtain accurate nonlinear models. To address these challenges, we developed Neural ODE-based Imitation Learning (NODE-IL), a novel model-based imitation learning framework that employs Neural Ordinary Differential Equations (Neural ODEs) for learning task dynamics and control policies. NODE-IL comprises (1) Dynamic-NODE for learning the continuous differentiable task's transition dynamics model, and (2) Control-NODE for learning a long-horizon control policy in an MPC fashion, which are trained holistically. Extensively evaluated on challenging manipulation tasks, NODE-IL demonstrates significant advantages in data efficiency, requiring less than 70 samples to achieve robust performance. It outperforms Behavioral Cloning from Observation (BCO) and Gaussian Process Imitation Learning (GP-IL) methods, achieving 70% higher average success rate, and reducing translation errors for high-precision tasks, which demonstrates its robustness and accuracy, as an effective and efficient imitation learning approach for learning complex manipulation tasks.**

I. INTRODUCTION

Learning physical skills has been a long-standing challenge in robotics, driven by the huge potential of deploying highly autonomous robots across industries. Recently, Reinforcement Learning (RL) serves as a powerful paradigm to acquire both manipulation and locomotion skills [1], [2] through trial-and-error to discover optimal policies. However, in addition to demanding computational resources, RL faces limited data efficiency and a high sensitivity to reward design, which makes alternative solutions more attractive where demonstration of desired tasks and behaviors can be reasonably accessible or available.

Learning through imitation – the ability to mimic and learn from an expert or teacher, is a fundamental aspect of human learning. Model-free imitation learning algorithms are capable of acquiring such abilities in numerous tasks, ranging from playing simple games [3], [4] to complex robot manipulation [5], [6] and locomotion [7], [8]. The simplest form of model-free imitation learning is behavior cloning (BC), which focuses on learning a mapping from state to action through a supervised learning [3], [4], [5], [6]. However, although the BC methods are efficient and adaptive, they may become problematic where the assumptions made during the training phase can quickly become outdated, leading to poor performance. Another typical model-free imitation learning algorithm is generative adversarial imitation learning (GAIL) [9], [10], [11], [12]. Despite of its robustness and generalization, GAIL methods require massive time and memory resources. This requirement stems from GAIL's reliance on an internal RL loop with random exploration.

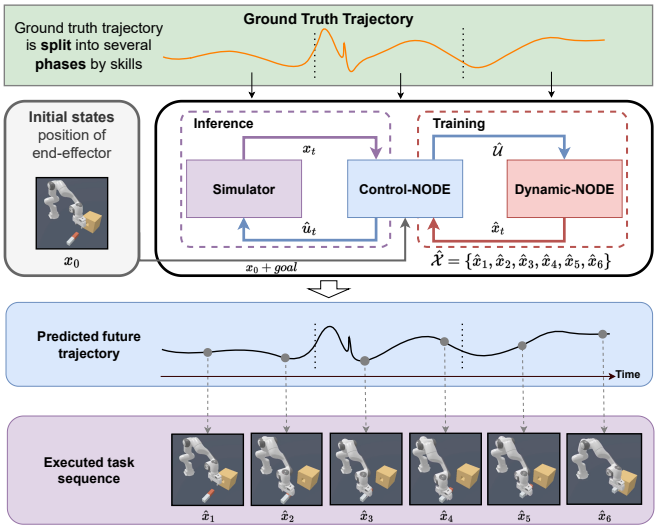

Fig. 1: NODE-IL has two Neural ODEs: (1) Dynamic-NODE for learning task dynamics, and (2) Control-NODE for learning a *long-horizon control policy*, using less than 70 trajectories to learn a robust policy for the target task.

Although BC-like methods such as Dataset Aggregation (DAgger) [13] and GAIL-like methods show great capacity for solving specific robot learning tasks, they predominantly focus on replicating the external behavior of the target agent and often ignore the underlying dynamics of specific tasks, which are crucial for effective control. Recent works have been focused on model-based imitation learning (MBIL). The performance of model-based imitation learning largely depends on the accuracy of the built model, however, obtaining an accurate model could be difficult, especially when dealing with a complex robotic system.

Many MBIL methods like model-based Gaussian process (GP) IL [14] and Behavioral Cloning from Observation (BCO) [3], have recently been pivotal in robotic manipulation. However, the limitations are magnified when it comes to learning high-precision tasks efficiently. High-precision tasks require precise control over movements and forces, which can be significantly affected by the discrete nature of traditional time-stepping methods. These methods

[1] Shiyao Zhao, Yucheng Xu, Mohammadreza Kasaei, and Mohsen Khadem are with the School of Informatics, University of Edinburgh, UK. Email: {shiyao.zhao, yucheng.xu, m.kasaei, mohsen.khadem}@ed.ac.uk

[2] Zhibin Li is with the Department of Computer Science, University College London, UK. Email: alex.li@ucl.ac.uk

This work is supported by the Medical Research Council [MR/T023252/1]

often struggle with capturing the complex dynamics and continuous variability inherent in real-world environments, leading to suboptimal performance in tasks that demand high levels of precision and adaptability.

To address these issues, we introduce Neural ODE-based Imitation Learning, NODE-IL, a novel model-based imitation learning framework that employs Neural Ordinary Differential Equations (Neural ODEs) [15], [16] for learning the task dynamics as well as the control policy. Specifically, NODE-IL comprises two main components: Dynamic-NODE for learning a continuous and differentiable task transition dynamics model and Control-NODE for learning a long-horizon control policy in a Model Predictive Control (MPC) fashion. Firstly, to capture the physical information embedded in the expert demonstration, we leverage the Dynamic-NODE to accurately and efficiently learn a continuous transition dynamics model for specific tasks. Secondly, the Control-NODE is designed to learn a long-horizon control policy by interacting with the pre-trained Dynamic-NODE, which serves as the system model. Benefiting from the *continuous*, *differentiable*, and *accurate* transition model learned by Dynamic-NODE and, the MPC strategy, the Control-NODE is able to efficiently learn accurate and robust control policy, which can significantly mitigate compound errors in long-horizon imitation learning tasks. The overall architecture of the proposed framework is depicted in Fig. 1.

Our contributions are as follows:

- We developed a novel model-based imitation learning framework, NODE-IL, which incorporates Neural ODEs consisting of Dynamic-NODE for learning a continuous and differentiable transition dynamics model of tasks, and Control-NODE for learning a long-horizon control policy in an MPC fashion.
- Training via this holistic formulation enables efficient learning of physical interactions with *three times fewer* demonstrations, compared to state-of-the-art BCO, GP-IL, BC and GAIL. NODE-IL generates optimal control sequences in an MPC fashion that effectively mitigates compound errors, resulting in more stable and robust performance.

II. Related Works

A. Imitation Learning

There are two main paradigms in imitation learning: model-free and model-based imitation learning. For example, the prevailing methods such as BC [17] and GAIL [10] are model-free imitation learning (MFIL) methods. These methods focus on imitating the behavior of experts without explicitly modeling the dynamics of the targeted tasks, requiring the collection of demonstration data for learning a category of skills [18], [19].

Additionally, MFIL methods generally suffering from data inefficiency and limited generalization beyond the types of skills of skill features that were not demonstrated before. Recent research [13], [20], [21] has been focused on model-based imitation learning (MBIL), which aims to leverage the physical information embedded in the expert demonstration and learn a model of task dynamics. Theoretically, MBIL methods are more efficient and generalizable as they can learn a compact representation of the system dynamics, which serves as a prior to enabling efficient policy learning and significantly improves the robustness and generalization of the learned policy. However, how to build an accurate model for the system dynamics still remains an open question, especially for tasks involving complex nonlinear dynamics (e.g. complex robot system with multiple objects).

B. Neural Ordinary Differential Equations

Stemming from the physical modeling of nonlinear dynamical system [22], [23], previous works [24] choose second-order ODE to represent intricate robot system. These methods solve the second-order ODE using analytical methods such as the Hamiltonian method [24], [25] and the Lagrangian method [26], [27], which may lead to high-dimensional state spaces that are computationally challenging to solve.

Recently, Neural Ordinary Differential Equations (Neural ODEs) was first proposed in [15], which interprets the forward pass of a ResNet [28] as solving a nonlinear ODE. The advent of Neural ODEs provides an efficient and effective way to model complex nonlinear dynamical systems, benefiting from adaptive time-stepping. The following work in [29] augmented the vanilla Neural ODEs by adding null space, which is capable of modeling any complex nonlinear dynamical system [30], [31]. Recently, Neural ODEs have been widely used in robotics research [21], [32], [33], showing great potential for modeling and controlling complex robot systems.

III. Methodology

A. Overview of NODE-IL

Our proposed NODE-IL framework has two primary components. First, a Neural ODE (i.e. **Dynamic-NODE**) is utilized to model a continuous task dynamics of a robotic system based on observed demonstrations under the Markov decision process (MDP) assumption: the next state only depends on the current state and the selected action. Basically, the Dynamic-NODE is trained to approximate the continuous and differentiable nonlinear ODE, which governs the task dynamics, by learning a function $f_{\theta_{\text{dyn}}}: \ x_t, u_t \rightarrow x_{t+1}$, where x_t and u_t stand for state and action at time t, respectively. Second, we employ another Neural ODE (i.e. **Control-NODE**) to learn a long-horizon control policy. The training objective of Control-NODE is to learn a function $f_{\theta_{\text{ctrl}}}: \ x_0, g \ \rightarrow \ \hat{\mathcal{U}}^N \ = \ \{u_0, \cdots, u_{N-1}\}$, where g stands for the goal condition, x_0 stands for the initial observation of the system, $\hat{\mathcal{U}}^N$ stands for the predicted N-step action sequence. The Dynamic-NODE and Control-NODE are trained in a mutually beneficial fashion. Specifically, Control-NODE gains from the precise state transition models generated by Dynamic-NODE, while Dynamic-NODE, in turn, benefits from the robust control policies formulated by Control-NODE. During the test phase, the Control-NODE directly

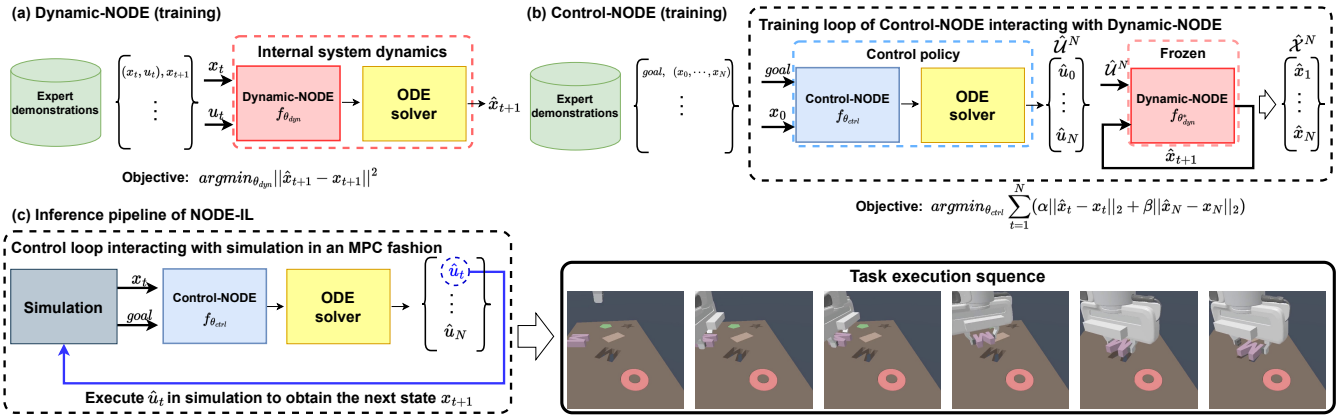

Fig. 2: The proposed method comprises two Neural ODEs: (a) learning the internal dynamics of the target task; (b) learning the MPC control strategy. (c) The inference pipeline generates control actions in an MPC fashion, which can use less than 70 trajectories and efficiently learn a robust policy for the target task.

interacts with the simulation and performs in MPC fashion. For state observation x_t, the Control-NODE receives $(x_t\ , g)$ and outputs $\{u_t, \cdots, u_{N-1}\}$ and only the first output (i.e. u_t) of Control-NODE will be fed into the simulation to control the agent. Overall, with all the main components and designs, our proposed NODE-IL is capable of achieving data-efficient imitation on complex robot manipulation tasks.

B. Dynamic-NODE

The target of our Dynamic-NODE is to learn the internal dynamics of the task under the control signals in a continuous-time fashion. However, as the control signal may vary over time, modeling long-horizon dynamics is difficult and may lead to accumulated errors. Thus, we made a basic assumption: *the control signal is static over a short time interval* τ. Based on this assumption, the target of our Dynamic-NODE is cast into modeling the single-step internal dynamics under the given control signals. To implement the Dynamic-Node, we adopt an augmented Neural ODE [16] as the Dynamic-NODE, which augments the vanilla Neural ODE by adding a null space. This augmented Neural ODE demonstrates superior capability in accurately modeling intricate dynamical systems. The architecture of Dynamic-NODE is presented in Fig. 2(a).

Let f be the nonlinear ordinary differential equation that governs the internal dynamics of the target system. A time-dependent multilayer perceptron (MLP) is used in our Dynamic-NODE as an estimator $f_{\theta_{\text{dyn}}}$, which approximates the target ordinary differential equation f by solving initial value problem (IVP), $\frac{\partial x(t)}{\partial t} = f_{\theta_{\text{dyn}}}(x(t), u_{t_i}, t)$, where the initial value $x(t) = x_t$ is the observation of the system state at time t, u_t stands for the input control signal. Then, the system state at time $t_{\text{i+1}}$ can be obtained by integrating the ordinary differential function, which is mathematically done by invoking a numerical ODE solver,

$$\begin{aligned}\hat{x}_{t_{\text{i+1}}} &= \text{ODESolver}(f_{\theta_{\text{dyn}}}, (x_{t_\text{i}}, u_{t_\text{i}}), (t_\text{i}, t_{\text{i+1}})) \\ &\simeq x_{t_\text{i}} + \int_{t_\text{i}}^{t_{\text{i+1}}} f_{\theta_{\text{dyn}}}(x_{t_\text{i}}, u_{t_\text{i}}, \tau) d\tau \end{aligned} \quad . \quad (1)$$

TABLE I: Training and model hyperparameters of Dynamic-NODE and Control-NODE

Hyperparameter	Value
Num. of hidden neuron ($f_{\theta_{\text{dyn}}}$)	320
Num. of hidden layer ($f_{\theta_{\text{dyn}}}$,$f_{\theta_{\text{ctrl}}}$)	3
Num. of hidden neuron ($f_{\theta_{\text{ctrl}}}$)	640
activation	ReLU&Tanh
optimizer	RMSProp
learning rate	1e-3
learning rate decay	0.1 per 100 epochs
weight decay	1e-4
batch size	16
total iterations	10k

In our framework, Runge-Kutta of Dormand-Prince [34] ODE solver is used to solve the Neural ODE together with the adjoint method [35] – more efficient and memory-friendly solving process. The accept tolerance and reject tolerance of the ODE solver are set to $1e-3$. The training objective of Dynamic-NODE is to minimize the mean squared errors (MSE loss) between predicted state $\hat{x}_{t_{\text{i+1}}}$ and its corresponding ground truth $x_{t_{\text{i+1}}}$. The training and model hyperparameters of Dynamic-NODE are listed in Table I.

C. Control-NODE

In order to leverage the learned Dynamic-NODE $f_{\theta_{\text{dyn}}}$, we design a model-based controller, Control-NODE, which is both data-efficient and robust with regard to system errors in the learned dynamics model. The target of Control-NODE in our proposed NODE-IL is to learn an optimized control policy π by interacting with the pre-trained Dynamic-NODE, which represents the task dynamics as a nonlinear differential equation parameterized by θ_{dyn}. Similar to the Dynamic-NODE, an augmented Neural ODE [16] is used to approximate the optimized control policy $f_{\theta_{\text{ctrl}}} \simeq \pi$.

The training process of Control-NODE is conducted iteratively over demonstration trajectories by interacting with the pre-trained Dynamic-NODE, as illustrated in Fig. 2(b). Given the initial state, x_0, and the goal condition, g, the Control-NODE predicts the action sequence, $\hat{\mathcal{U}}^N = \{u_0, \cdots, u_{N-1}\}$ over a predefined finite horizon, N, subject to Dynamic-NODE ($f_{\theta_{\text{dyn}}}: \ x_t, u_t \rightarrow x_{t+1}, $). Then, each action in $\hat{\mathcal{U}}^N$ is fed into the Dynamic-NODE together

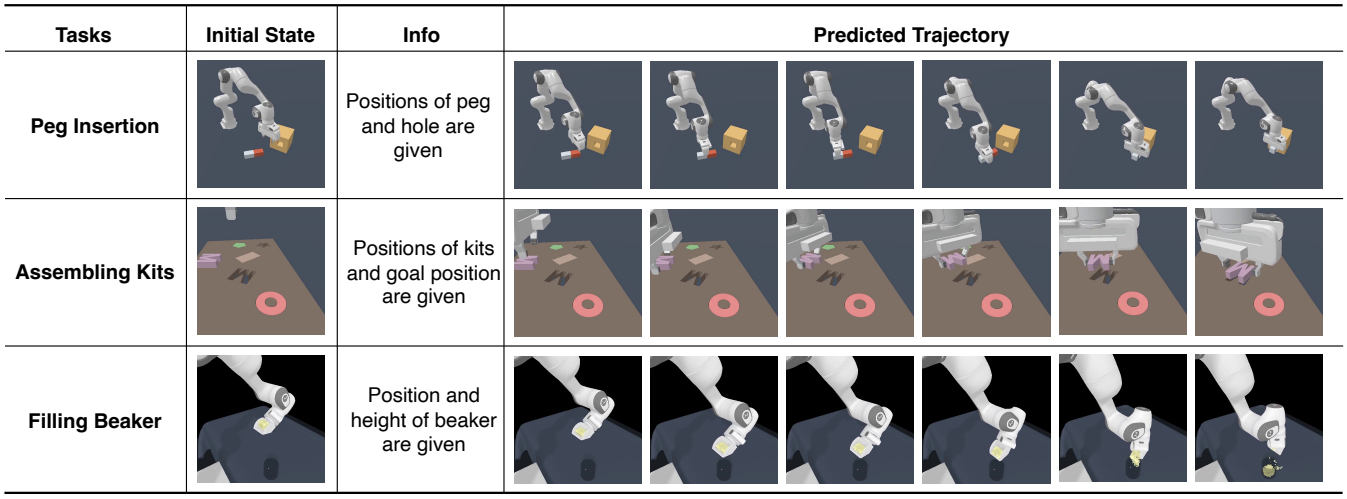

Fig. 3: Evaluation validated on three high-precision tasks. Each task provides task-related information as the initial condition, and then NODE-IL model predicts the trajectory to achieve the goal.

with the current state prediction to generate the estimation of the next state iteratively, and is mathematically written as:

$$\begin{aligned}
&\hat{\mathcal{U}}^N = \text{ODESolver}(f_{\theta_{\text{ctrl}}}, x_0, g, (t_0, \cdots, t_{N-1})), \\
&\text{For } \ \hat{u}_t \in \hat{\mathcal{U}}^N, \ \ \hat{x}_{t+1} = \text{Dynamic-NODE}(x_t, \hat{u}_t), \\
&\text{where, } \ x_t = \begin{cases} x_0, & \text{if } \ t = 0 \\ \text{Dynamic-NODE}(x_{t-1}, \hat{u}_{t-1}), & \text{if } \ t > 0 \end{cases}.
\end{aligned} \tag{2}$$

Therefore, after iterating over $\hat{\mathcal{U}}^N$, the output action sequence of Control-NODE is translated into a state sequence $\hat{\mathcal{X}}^N = \{\hat{x}_1, \cdots, \hat{x}_N\}$. The training objective of Control-NODE is to minimize the pair-wise MSE loss between the predicted state sequence $\hat{\mathcal{X}}^N$ and their corresponding ground truth $\mathcal{X}^N$, plus a terminal cost term. α and β serve as weighting coefficients that indicate the relative significance of each loss component. In our setting, α is set to 500, and β is set to 100.

$$\underset{\theta_{\text{ctrl}}}{\text{argmin}}(\underbrace{\alpha||\hat{\mathcal{X}}^N - \mathcal{X}^N||_2}_{\text{MSE loss}} + \underbrace{\beta||\hat{x}_N - x_N||_2}_{\text{terminal cost}}) \tag{3}$$

IV. Experiments

In this section, we first introduce the tasks we used to evaluate our proposed NODE-IL, including rigid-body manipulation and soft-body manipulation. Based on the level of control precision required by these tasks, they can be further divided into high-precision tasks and normal-precision tasks. Then, we present our data collection and processing pipeline. After that, we present both visual results and statistical results of our NODE-IL on these tasks. Moreover, we present the comparison between NODE-IL against the BC and GAIL methods in terms of both successful rate and precision.

A. Tasks

We evaluate NODE-IL on five robot manipulation tasks with a Franka robot in the SAIPEN physics simulation engine [36], including: Peg Insertion, Kits Assembling, Beaker Filling, Hanging, and Excavating. These tasks can be categorized into high-precision tasks (Peg Insertion, Kits Assembling, Beaker Filling) and normal-precision tasks (Hanging, and Excavating). For high-precision tasks, the robots need to reach the goal condition with a small error tolerance while in the normal-precision tasks, the error tolerance is bigger. Therefore, the translation error, rotation error, and success rate are used as the evaluation metrics for high-precision tasks, while only the success rate is used for normal-precision tasks. Moreover, to demonstrate the generalization of our proposed NODE-IL, both soft-object manipulation (Beaker Filling, Hanging, and Excavating) and rigid-object manipulation (Peg Insertion, Kits Assembling) tasks are included. Details of each task are introduced in the following paragraphs:

Peg Insertion. The goal of this task is to successfully insert a peg into the hole in the target box. The dimension of the peg's insertion face is $25 \times 25mm$, while the dimension of the hole is $34 \times 34mm$, meaning the error tolerance during insertion is $9mm$ along each axis. The initial state of this task includes 3D positions of the end-effector, peg position, and target hole position.

Kits Assembling. The success criteria of the task is to determine whether the kit is placed in the corresponding target position while the shift between object and goal in the X-Y plane is smaller than $20mm$, and the orientation shift is smaller than $10°$. The initial state of this task includes 3D positions of the end-effector and the position pair of the kit and its target.

Beaker Filling. The task is to fill soft materials (e.g. clay) into a beaker. The task success is assessed by the percentage of soft materials filling inside the beaker, and we set this clearance threshold to 90%. The initial state of this task includes 3D positions of the end-effector and beaker.

Excavating. Normal-precision task. The objective of the task is to scoop up a certain amount of soft materials. Instead of specifying the number of particles, we define the depth that the end-effector would dig as the criteria. It would be

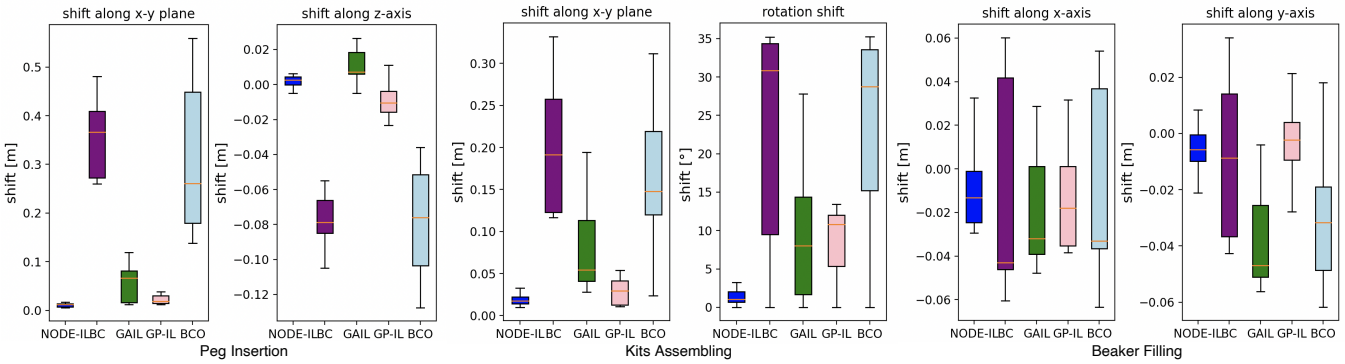

Fig. 4: The translation error and rotation error in high-precision tasks. The translation error is calculated as a shift between the center of objects and the corresponding target along a specific axis. The rotation error is the axis-angle between the object and the target. The kits assembling task is sensitive to rotation error.

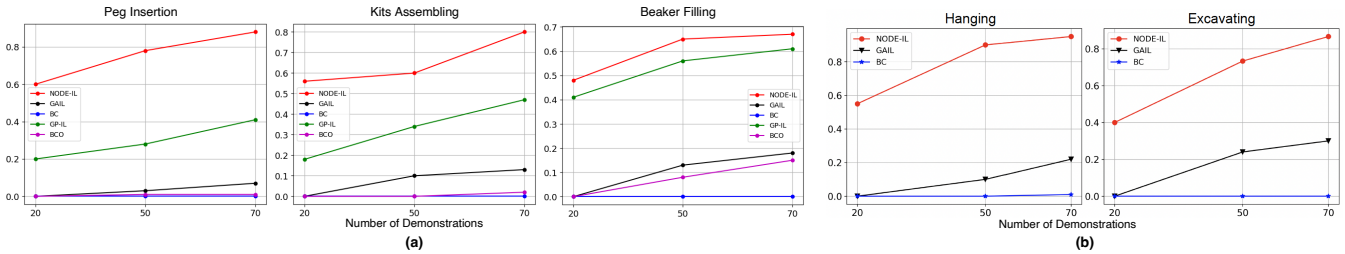

Fig. 5: The success rate with different numbers of demonstrations over five manipulation tasks. Our model outperforms GP-IL, BCO, BC and GAIL. The average success rate on normal-precision tasks is higher than that on high-precision tasks.

considered as a successful trial if the end-effector digs deeper than a specific depth. The initial state, thus, includes the position of the end-effector and pool, as well as the depth information that the end-effector is about to reach.

Hanging. Normal-precision task. The goal of this task is to hang a towel onto a rod safely and stably. The success criteria of this task is that the towel can stay on the rod when robot drops off the towel. As the task does not involve grasping the towel, the initial state only includes the 3D positions of the end-effector and the target rod.

B. Data Collection and Processing

The raw expert demonstrations are generated by performing state-based Reinforcement Learning using implementations from stable-baselines3 [37] benchmark. We first extract the action sequence, which contains the positions of all robot joints, from each expert demonstration. Then, to reduce the dimension and increase the computational efficiency, each action in an action sequence is converted to a Tool Center Point (TCP) including the 3D position and rotation of the end-effector, which is written as $u_t^{TCP} = \{p_t, R_t, \psi\}$, where $p_t = (x_t, y_t, z_t)$ denotes the 3D position of the end-effector, the $R_t = (\alpha, \beta, \gamma)$ denotes the 3D rotation of the end-effector, and the ψ stands for a binary identifier which has value -1 or 1, identifying the state of gripper, where -1 means the gripper is closed and 1 means opened. The ψ is ignored in tasks that do not involve the grasping process. Notably, the p_t and R_t are further normalized to range $[-1, 1]$ by the conversion tool provided by ManiShill2 benchmark [38]. The state sequence is a Box space carrying TCP, object and target positions.

Since our targeted tasks are long-horizon and require multiple skills, we adopt a skill-decoupling strategy to decouple the whole demonstration sequence into segments that can be completed by individual skills. For completing a multi-skill task, we train separate NODE-ILs for each skill. For example, for the peg insertion and assembling kits task, we split these two tasks into three sub-tasks: (1) reaching, (2) grasping, and (3) reaching for the target. In order to precisely split the expert demonstrations, we conduct two different strategies for tasks with and without the grasping process. For those tasks with a grasping process (e.g. peg insertion and assembling kits), we split the demonstration sequence according to the state of the gripper, as the state of the gripper will switch from 1 to -1 when the robot starts to grasp the target object. For those tasks without a grasping process (e.g. Beaker Filling, hanging, and excavating), the distance between the end-effector and the target position is used to split the raw demonstration sequence. The snapshots and information of each task are presented in Fig. 3.

C. Inference with NODE-IL

The inference pipeline of NODE-IL is illustrated in Fig. 2(c). As we mentioned in Sections III-A and III-C, during the test phase, the Control-NODE directly interacts with the simulation environment in an MPC fashion, which results in a most robust performance. Given the current state x_t and goal condition g, which can be obtained from the simulator, the Control-NODE predicts a finite action sequence $\mathcal{U}_t^N = \{u_t, \cdots, u_{t+N-1}\}$. Then, only the first action item $u_t = (p_t, R_t, \psi)$ is fed into the simulator. Then, an internal controller, which is provided by the ManiSkill2

TABLE II: Comparison of success rate on all five tasks.

Model	Peg Insertion	Kits Assembling	Beaker Filling	Excavating	Hanging
BC [38]	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.02
GAIL [38]	0.07 ± 0.02	0.13 ± 0.02	0.18 ± 0.03	0.3 ± 0.07	0.22 ± 0.12
BCO [3]	0.01 ± 0.01	0.02 ± 0.01	0.14 ± 0.02	–	–
GP-IL [39]	0.47 ± 0.01	0.41 ± 0.02	0.61 ± 0.02	–	–
NODE-IL	**0.88** ± 0.02	**0.80** ± 0.03	**0.67** ± 0.04	**0.94** ± 0.03	**0.95** ± 0.02

TABLE III: Comparison of translation errors on High-precision tasks.

Model	Peg Insertion	Kits Assembling	Beaker Filling
BC [38]	$0.31m \pm 0.08$	$0.22m \pm 0.03$	$0.24m \pm 0.07$
GAIL [38]	$0.089m \pm 0.022$	$0.046m \pm 0.012$	$0.065m \pm 0.013$
BCO [3]	0.30 ± 0.11	0.17 ± 0.05	0.051 ± 0.013
GP-IL [39]	0.025 ± 0.009	0.028 ± 0.012	0.027 ± 0.009
NODE-IL	**0.016m** ± 0.003	**0.018m** ± 0.007	**0.017m** ± 0.008

TABLE IV: Dynamic GP-IL and Dynamics-NODE comparisons on Peg Insertion and Kits Assembling tasks. E_T denotes the trajectory translation error in meter, E_R denotes the rotation error in degree, and R_S denotes the success rate.

Dynamics Model	Peg Insertion			Kits Assembling			
	$E_T{}^m\downarrow$	$E_R{}^\circ\downarrow$	$R_S\uparrow$	$E_T{}^m\downarrow$	$E_R{}^\circ\downarrow$	$R_S\uparrow$	T_{train}
GP-IL	0.018 ± 0.002	2.3 ± 0.8	0.47 ± 0.01	0.022 ± 0.004	4.3 ± 1.8	0.41 ± 0.02	$3h$
NODE	**0.016** ± 0.003	**2.3** ± 1.1	**0.88** ± 0.02	**0.018** ± 0.007	**4.2** ± 2.7	**0.80** ± 0.03	**45 mins**

benchmark converts the input p_t and R_t to the positions of all robot joints, which can be directly executed by the simulator. The state of the robot system at the next timestep $t+1$ can be observed after executing the u_t. By iterating the above steps, the Control-NODE can predict an optimized trajectory for achieving the goal condition.

D. Quantitative Results and Comparison

To statistically demonstrate the advancement of our proposed NODE-IL, we conducted extensive simulations on the targeted tasks mentioned in Section IV-A and compared our NODE-IL with GP-IL, BCO, BC and GAIL methods. Notable, the official implementations of BC and GAIL from ManiSkill2 are used for training [38]. The GAIL implementation in ManiSkill2 is improved by incorporating the Soft-Actor Critic (SAC) [40] to provide dense reward. We also modified the GP-IL by plugging dynamics GP into NODE-IL architecture and replacing Dynamic-NODE. To ensure a fair comparison, the size of the training dataset is consistent across all methods. For each task, we collect 100 trajectories and randomly split them into the training set (70 trajectories) and the testing set (30 trajectories). We first train our NODE-IL on the training set with fixed 10k iterations and then evaluate it on the testing set to calculate the quantitative results. This process is repeated five times for each task with different random train-test splits, and the mean values of each metric are reported. Furthermore, to examine the performance and efficiency of NODE-IL due to Dynamic-NODE, we conducted evaluation of Dynamic-NODE against Dynamic GP.

The quantitative results, including the success rates on five tasks, and the translation errors on the high-precision tasks, are demonstrated in Table II and Table III, respectively. It can be seen that NODE-IL significantly outperforms the selected MFIL and MBIL method [38] in terms of both success rate and translation errors on all challenging tasks (see Fig. 4 and Fig. 5). Specifically, NODE-IL can reach an overall 80% success rate on high-precision tasks and over 90% on normal-precision tasks, while the BC and BCO method nearly failed on all tasks, and the GAIL method only reaches an overall 17.4% success rate. Although GP-IL achieves around 50% average success rate on three high-precision tasks and offers better stability in dynamic modeling, it requires dense computational resources. The performance analysis in Table IV shows that Dynamic-NODE is 75% more efficient than GP-IL in terms of computational efficiency.. Additionally, to illustrate the data efficiency of our proposed NODE-IL and compare it against the BC, GAIL, BCO and GP-IL method, we present plots in Fig. 5 that present success rates of NODE-IL, BC, and GAIL against the number of demonstrations used for training. As shown in Fig. 5, our method is capable of efficiently learning a robust control policy from only 70 demonstrations and reaching an average success rate of 85% over five tasks, which surpasses the BC method and GAIL method with a significant margin. Moreover, we observed that NODE-IL has a non-zero success rate when the number of demonstrations is 20, while the success rate of BCO, BC and GAIL methods drops to zero. Notably, we also observed that GAIL can reach a success rate close to our NODE-IL only by learning from more than ten times demonstrations (1000 vs. 70), meaning that our method has superior data efficiency.

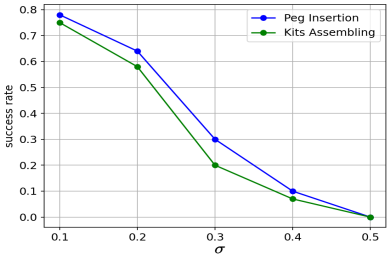

Fig. 6: Success rate of Peg Insertion and Kits Assembling tasks over increasing noise levels.

TABLE V: Results of ablation study on Peg Insertion and Kits Assembling tasks.

Method	Peg Insertion			Kits Assembling		
	$E_T{}^m\downarrow$	$E_R{}^\circ\downarrow$	$R_S\uparrow$	$E_T{}^m\downarrow$	$E_R{}^\circ\downarrow$	$R_S\uparrow$
NODE-IL	**0.016** ± 0.003	**2.3** ± 1.1	**0.88** ± 0.02	**0.018** ± 0.007	**4.2** ± 2.7	**0.82** ± 0.04
- *w/o Dynamic-NODE*	0.021 ± 0.008	3.5 ± 1.1	0.84 ± 0.03	0.020 ± 0.008	5.1 ± 3.1	0.75 ± 0.09
- *w random noise* $\sigma = 0.1$	0.016 ± 0.004	2.6 ± 1.2	0.78 ± 0.04	0.020 ± 0.006	5.4 ± 3.1	0.70 ± 0.05
- *w random noise* $\sigma = 0.2$	0.018 ± 0.009	2.8 ± 1.7	0.64 ± 0.04	0.19 ± 0.005	6.1 ± 3.9	0.58 ± 0.02

V. Ablation Study

We conducted an ablation study on peg insertion and kits assembling tasks to justify (i) the effectiveness of the learned Dynamic-NODE as well as (ii) demonstrate the robustness of our NODE-IL: We add random gaussian noise $\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2)$ on the state observations from simulation during the testing phase, where $\mu = 0$. We conducted experiments with $\sigma = [0.1, 0.2, 0.3, 0.4, 0.5]$, separately on selected high-precision tasks. The quantitative results of the ablation study are presented in Table V and Fig. 6. The results of (*i*) show that interacting with a learned dynamics model can improve the performance of NODE-IL by 5.5 percent, which supports the effectiveness and necessity of a learned dynamics model. Moreover, the results of (*ii*) show

that our NODE-IL is able to learn a control policy that is robust to random noise when σ is smaller than 0.2. Overall, our ablation study supports the framework design and robustness of NODE-IL.

VI. Conclusion

This work presents a novel model-based imitation learning framework, NODE-IL, leveraging Neural ODEs to learn task dynamics and control policies. Our framework consists of (1) Dynamic-NODE for learning the continuous task transition dynamics from expert demonstrations, and (2) Control-NODE for learning a long-horizon control policy and interacting with the environment in an MPC fashion.

Results from extensive experiments demonstrate the advancement of our NODE-IL on five challenging robot manipulation tasks, showing that our NODE-IL can efficiently learn robust control policy from less than 70 demonstrations, which is ten times less than previous MFIL methods. Moreover, our NODE-IL significantly surpasses the BCO, GP-IL methods in terms of success rate (50% higher on average) and translation errors (60% improvement). In future work, we will explore NODE-IL's scalability to more complex tasks, such as multi-agent interactions and dynamic environments, while integrating with reinforcement learning and unsupervised methods to enhance data efficiency and adaptability. We also aim to improve computational efficiency and real-time adaptability, and work on the sim-to-real transfer for real-world deployments.

References

[1] Z. Sun, *et al.*, "Learning pregrasp manipulation of objects from ungraspable poses," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9917–9923.

[2] N. Hansen, *et al.*, "TD-MPC2: Scalable, Robust World Models for Continuous Control," in *International Conference on Learning Representations (ICLR)*, 2024.

[3] F. Torabi, *et al.*, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[4] A. Edwards, *et al.*, "Imitating latent policies from observation," in *International conference on machine learning*. PMLR, 2019, pp. 1755–1763.

[5] T. Zhang, *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 5628–5635.

[6] Y. Liu, *et al.*, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1118–1125.

[7] X. B. Peng, *et al.*, "Learning agile robotic locomotion skills by imitating animals," *arXiv:2004.00784*, 2020.

[8] J. Nakanishi, *et al.*, "Learning from demonstration and adaptation of biped locomotion," *Robotics and autonomous systems*, vol. 47, no. 2-3, pp. 79–91, 2004.

[9] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[10] F. Torabi, *et al.*, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[11] Y. Li, *et al.*, "Infogail: Interpretable imitation learning from visual demonstrations," *Advances in neural information processing systems*, vol. 30, 2017.

[12] J. Fu, *et al.*, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv:1710.11248*, 2017.

[13] S. Ross, *et al.*, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[14] N. D. Ratliff, *et al.*, "Gaussian process imitation learning," in *Advances in neural information processing systems*, 2009.

[15] R. T. Chen, *et al.*, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[16] E. Dupont, *et al.*, "Augmented neural odes," *Advances in neural information processing systems*, vol. 32, 2019.

[17] J. Ho, *et al.*, "Model-free imitation learning with policy optimization," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, USA: PMLR, 20–22 Jun 2016, pp. 2760–2769.

[18] S. Wang, *et al.*, "Learning adaptive grasping from human demonstrations," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 3865–3873, 2022.

[19] E. Triantafyllidis, *et al.*, "Hybrid hierarchical learning for solving complex sequential tasks using the robotic manipulation network roman," *Nature Machine Intelligence*, vol. 5, no. 9, pp. 991–1005, 2023.

[20] A. Hu, *et al.*, "Model-based imitation learning for urban driving," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20 703–20 716, 2022.

[21] H. Lin, *et al.*, "No need for interactions: Robust model-based imitation learning using neural ODE," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 088–11 094.

[22] B. Chang, *et al.*, "Antisymmetricrnn: A dynamical system view on recurrent neural networks," *arXiv preprint arXiv:1902.09689*, 2019.

[23] S. H. Strogatz, *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[24] Y. Feng, *et al.*, "Model-based reinforcement learning with a hamiltonian canonical ode network," 2022.

[25] K. Qian and L. Tian, "Data-driven physical law learning model for chaotic robot dynamics prediction," *Applied Intelligence*, vol. 52, no. 10, pp. 11 160–11 171, 2022.

[26] M. Lutter, *et al.*, "Deep lagrangian networks: Using physics as model prior for deep learning," in *International Conference on Learning Representations*, 2018.

[27] S. Y. Naing and T. Rain, "Analysis of position and angular velocity of four-legged robot (mini-bot) from dynamic model using euler-lagrange method," in *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*. IEEE, 2019, pp. 1–4.

[28] K. He, *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] R. Jena, *et al.*, "Augmenting gail with bc for sample efficient imitation learning," in *Conference on Robot Learning*. PMLR, 2021, pp. 80–90.

[30] T. Teshima, *et al.*, "Universal approximation property of neural ordinary differential equations," *arXiv preprint arXiv:2012.02414*, 2020.

[31] H. Zhang, *et al.*, "Approximation capabilities of neural odes and invertible residual networks," in *International Conference on Machine Learning*, 2020.

[32] M. Kasaei, *et al.*, "Data-efficient non-parametric modelling and control of an extensible soft manipulator," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2641–2647.

[33] Z. Meleshkova, *et al.*, "Application of neural ode with embedded hybrid method for robotic manipulator control," *Procedia Computer Science*, vol. 193, pp. 314–324, 2021.

[34] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[35] L. S. Pontryagin, *Mathematical theory of optimal processes*. CRC press, 1987.

[36] F. Xiang, *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 097–11 107.

[37] A. Raffin, *et al.*, "Stable-baselines3: Reliable reinforcement learning implementations," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 12 348–12 355, 2021.

[38] J. Gu, *et al.*, "Maniskill2: A unified benchmark for generalizable manipulation skills," 2023.

[39] P. Englert, *et al.*, "Probabilistic model-based imitation learning," *Adaptive Behavior*, vol. 21, pp. 388 – 403, 2013.

[40] T. Haarnoja, *et al.*, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.