Domain Adaptation with Hybrid Modeling for Learning Dynamical Systems

Akira Osaka AKR-OSAKA@G.ECC.U-TOKYO.AC.JP
Naoya Takeishi NTAKE@G.ECC.U-TOKYO.AC.JP
Takehisa Yairi YAIRI@G.ECC.U-TOKYO.AC.JP
The University of Tokyo

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Domain shifts present significant challenges for data-driven modeling of dynamical systems, as they may reduce state prediction accuracy and degrade model-based control performance. Transfer learning is a promising way to mitigate the effect of changes in dynamics. In this study, we investigate a domain adaptation framework based on hybrid modeling with fine-tuning. Hybrid models integrate physics-based components derived from prior knowledge of system dynamics into neural ordinary differential equations (neural ODEs). They are expected to facilitate efficient and enhanced fine-tuning because the structures of physics parts often remain invariant under domain shifts. We evaluated the hybrid neural ODE approach through experiments on multicopters undergoing concept shifts and found that introducing physics models significantly enhanced the domain adaptation capabilities, even when the physics-based components included unidentified parameters. Moreover, the results demonstrated that the hybrid modeling strategy reduced the amount of data required in the target domain, enabling efficient domain adaptation.

Keywords: Hybrid modeling; Domain adaptation; Dynamical system modeling

1. Introduction

Accurately modeling dynamical systems is essential, as model-based control methods such as model predictive control (MPC) heavily depend on the underlying system representation (Hewing et al., 2020b). In robotics, precise system identification and optimal control of dynamical systems are critical, as they directly influence safety and efficiency. While theoretical models derived from the prior knowledge of physics can only describe the limited part of the real-world dynamics (Brunke et al., 2022), data-driven modeling offers an alternative direction, enabling system identification directly from observed data. It allows models to capture complex real-world phenomena more effectively.

Among various data-driven modeling techniques, neural ordinary differential equations (neural ODEs) (Chen et al., 2018) is one of the notable approaches. In neural ODEs, neural networks are employed to learn the time derivative of system states and enable continuous-time modeling of dynamical systems (Chen et al., 2018), which means the function $f(x)$ in the equation of motion $\dot{x} = f(x)$ can be directly learned without recurrent network architectures. This formulation is advantageous, as we can naturally combine them with established control techniques (e.g., MPC, H_∞ control), which typically assume dynamics in the form of $\dot{x} = f(x, u)$.

Despite the effectiveness of data-driven models in capturing real-world behaviors, in practice they often face a major challenge of domain shift. Domain shift refers to distributional changes from source to target domains, which is problematic because most machine learning methods assume that the training and test data come from the same distribution. After the distributional shifts, models trained with the source domain data do not generalize well to the target ones (Singhal et al., 2023; Farahani et al., 2021). Since collecting new data from scratch in the target domain is often costly and time-consuming, it is essential to develop domain adaptation methods that allow source-trained models to be efficiently adapted to the target domain with limited labeled data.

A key strategy for domain adaptation is to incorporate shared features across source and target domains, as domain-invariant features are considered to facilitate the effective transfer of the source-trained models to target environments (Singhal et al., 2023), such as meta-learning approach in Wang et al. (2022); Kirchmeyer et al. (2022); Nzoyem et al. (2025). Regarding robotic systems, a distinguishing characteristic compared with other machine learning problems such as computer vision and natural language processing is the availability of theoretical physics-based models (e.g., equations of motion). Although the physics models may have some deviations from the real-world dynamics, their structural formulations are generally consistent across domains. Motivated by this insight, we suggest that the integration of physics models into neural ODEs enhances the domain adaptation capabilities of dynamical system models.

The approach of combining theoretical and data-driven models is known as *hybrid (or grey-box) modeling*. Previous studies have shown that incorporating physics-based components improves modeling performances, even when these physics parts contain unidentified parameters or modeling errors (Yin et al., 2021; Takeishi and Kalousis, 2021). It suggests that even imperfect physics models help to achieve better generalization to unseen data. However, the existing work on hybrid modeling often assumes that training and test data are drawn from the same distributions, and the application of these models to the domain adaptation problems has been explored much less.

In this study, we demonstrate that hybrid modeling contributes to enhancing the state prediction capabilities of dynamical systems undergoing domain shifts, even when only incomplete physics models are available. Specifically, we investigate the fine-tuning-based transfer, where hybrid neural ODEs are pre-trained using source domain data and then fine-tuned with a limited amount of target domain data, as illustrated in Fig. 1. We hypothesize that introducing physics components leads to more efficient fine-tuning because the physics structures work as the domain-invariant features. To evaluate our approach, we conducted experiments on multicopters experiencing changes in physics parameters and environments (i.e., domain shifts). We empirically found that incorporating physics-based components enhanced the prediction performance of neural ODEs and enabled few-shot learning with reduced data requirements.

2. Related work

2.1. Domain adaptation

Domain adaptation in the presence of distributional shifts is a crucial issue for data-driven models, as such shifts can severely degrade their prediction performance (Wang et al., 2021).

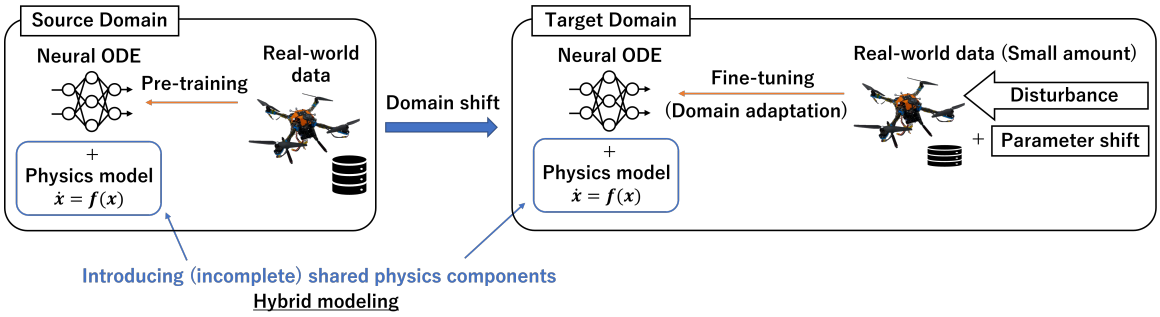

Figure 1: Hybrid modeling approach for domain adaptation with fine-tuning.

While fine-tuning is a typical domain adaptation approach, the recent work has focused on meta-learning, in which common features across multiple environments are extracted by neural networks for the fast adaptation to unseen domains (Wang et al., 2022; Kirchmeyer et al., 2022; Nzoyem et al., 2025). Although meta-learning is a promising strategy for domain adaptation, it typically assumes access to multiple domains (i.e., pairs of support and query sets) in the meta-training phase, whereas our setting assumes access to only a single pair of source and target domains. Conducting meta-learning only with two support-query pairs would be challenging. Therefore, we tackle the problem from a different perspective: utilizing physics knowledge to facilitate domain adaptation rather than attempting to learn such knowledge from scratch.

Hybrid modeling to address distribution shifts has been explored in Wehenkel et al. (2023), where they proposed Expert Augmentation, a data augmentation approach to improve generalization to out-of-distribution data by utilizing physics models. In contrast to their scenario, which assumes no access to target data after distribution shifts, we suppose that a limited amount of target-domain data is available for fine-tuning. Our objective is to demonstrate that hybrid models benefit from the physics-based components, whose structure should remain consistent across source and target domains, and enable more efficient fine-tuning compared to standard neural ODEs.

2.2. Hybrid modeling

The integration of physics knowledge into data-driven models has been widely studied, and various hybrid modeling approaches have been proposed across a broad range of research fields. One common direction in hybrid learning is to predict variables in dynamical system models, such as parameter estimation in robotics (Heiden et al., 2021), and covariate estimations in epidemiology (Arik et al., 2020) and meteorology (Verma et al., 2024). Hamiltonian neural networks (Greydanus et al., 2019) are another well-known approach, incorporating a structure of analytical mechanics into models.

Another class of hybrid models is residual learning, in which data-driven models are used to compensate for discrepancies between real-world data and physics models obtained in advance. APHYNITY in Yin et al. (2021) is a notable example of this residual hybrid model, which demonstrated that even incomplete physics models with unidentified parameters can significantly improve prediction accuracy. Residual forms of hybrid modeling have also been

applied in generative modeling (Takeishi and Kalousis, 2021) and robotics (Bauersfeld et al., 2021). Takeishi and Kalousis (2023) and Osaka et al. (2025) investigated the formulation of loss functions to fully exploit available prior physics knowledge. In addition, we can find the application to robotics system control such as MPC based on hybrid models (Hewing et al., 2020a; Torrente et al., 2021; Salzmann et al., 2023). However, domain adaptation with fine-tuning in the context of hybrid modeling remains underexplored so far.

3. Methods

3.1. Problem setting

We consider scenarios where dynamical system models trained on source domain data are available, but due to changes in system dynamics, these models must be adapted to maintain performance in a target domain. Particularly we focus on so-called *concept shifts*:

$$p_{\mathrm{s}}(y \mid x) \neq p_{\mathrm{t}}(y \mid x),\ p_{\mathrm{s}}(x) = p_{\mathrm{t}}(x), \tag{1}$$

where x denotes input variables, y denotes target values, and p_{s} and p_{t} denote the data distributions in source and target domains, respectively (see, e.g., Moreno-Torres et al., 2012). Our goal is to obtain a model that predicts the trajectories of dynamical systems accurately in the target domain. Besides, we aim to reduce the required amount of target data, i.e., few-shot learning in the target domain.

3.2. Hybrid modeling

The neural ODEs for dynamical systems can be written as

$$\dot{x} = f_{\mathrm{d}}(x, u; \theta_{\mathrm{d}}), \tag{2}$$

where f_{d} is the neural network parameterized by θ_{d}, which takes the state variable x and control input u as the input. The notation $\dot{x}$ means the time derivative of x. Note that for modeling robot systems, control input u is considered in addition to the standard form of neural ODEs (Chen et al., 2018).

We assume that we have prior knowledge of the dynamics derived from our theoretical understanding of the underlying physics in a form:

$$\dot{x} = f_{\mathrm{p}}(x, u; \theta_{\mathrm{p}}). \tag{3}$$

It is the equation of motion parametrized by unknown parameters θ_{p}, which may be different between source and target domains. Although the parameters θ_{p} may vary, the structural form of the physics part remains the same, and our idea is to utilize this invariant structure of the equations for addressing distributional shifts. The physics component should be determined based on the fundamental scientific understanding of the dynamics underlying data.

Importantly, the physics equations need not perfectly match the actual dynamics. Some parameters may remain unidentified, and certain terms may be omitted. The idea of hybrid modeling is to compensate such deviation by combining the physics model with a data-driven

model. More specifically, we employ the additive formulation of hybrid models developed in Yin et al. (2021):

$$\dot{x} = f_{\mathrm{p}}(x, u; \theta_{\mathrm{p}}) + f_{\mathrm{d}}(x, u; \theta_{\mathrm{d}}). \tag{4}$$

The equation of motion, f_{p}, can often be regarded as a domain-invariant structure (modulo unknown θ_{p}) because the physics laws do not change between domains. We suggest that incorporating the physics models into the neural ODEs as hybrid modeling is beneficial for domain adaptation.

3.3. Pre-training in source domain

We first learn the parameters of the hybrid model (θ_{p} and θ_{d}) with sufficient amount of source domain data. Pre-training with the source data is performed as with the standard neural ODE training procedure. The dataset in the source domain is

$$\begin{gathered}\mathcal{D}_{\mathrm{s}} \coloneqq \{(x_0, \boldsymbol{u}, \boldsymbol{y})\},\ \boldsymbol{u} = [u_0, u_1, \ldots, u_T],\ \boldsymbol{y} = [x_0, x_1, \ldots, x_T], \\ \text{where } \dot{x}_t = f(x_t, u_t) \text{ for } t \in [0, T].\end{gathered} \tag{5}$$

$\boldsymbol{y}$ is the trajectory to be predicted from the initial state x_0 and the sequence of inputs $\boldsymbol{u}$, under the dynamics $\dot{x} = f(x, u)$.

Given $(x_0, \boldsymbol{u})$ as an input, we can integrate the hybrid model in Eq. (4) along time t and obtain a predicted trajectory $\tilde{\boldsymbol{y}} = [x_0, \tilde{x}_1, \ldots, \tilde{x}_T]$. By minimizing the distance between the prediction $\tilde{\boldsymbol{y}}$ and the label $\boldsymbol{y}$, the hybrid model is trained. We adopt the mean squared error (MSE) as the loss function here: $\mathcal{L}_{\mathrm{MSE}}(\theta_{\mathrm{p}}, \theta_{\mathrm{d}}) = \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|_2 / T$.

By merely minimizing $\mathcal{L}_{\mathrm{MSE}}$, both parameters θ_{p} and θ_{d} are moved freely without constraints. For this learning scheme, several previous studies (Yin et al., 2021; Takeishi and Kalousis, 2023) point out that a constraint (or a soft constraint as a regularizer) on the data-driven part f_{d} is required, as the data-driven models such as neural networks are highly flexible, and they can overwrite the physics part f_{p}, resulting in the poor estimation of unknown physics parameters θ_{p}. Thus, we adopt the regularized training approach by adding a regularizer to the MSE loss $\mathcal{L}_{\mathrm{MSE}}$. The regularized loss function is

$$\mathcal{L}(\theta_{\mathrm{p}}, \theta_{\mathrm{d}}) = \mathcal{L}_{\mathrm{MSE}} + \lambda \cdot \mathcal{L}_{\mathrm{Reg}}, \tag{6}$$

where $\mathcal{L}_{\mathrm{Reg}}$ is a regularizer, and λ is the balancing hyperparameter. We will denote the optimized parameters by $\theta_{\mathrm{p}}^*, \theta_{\mathrm{d}}^* = \arg\min_{\theta_{\mathrm{p}}, \theta_{\mathrm{d}}} \mathcal{L}$. We use a correlational regularizer to increase the similarity between the physics model f_{p} and the whole model $f_{\mathrm{p}}+f_{\mathrm{d}}$ to indirectly limit the output of the data-driven component, as investigated in Osaka et al. (2025). The correlational regularizer is defined as

$$\mathcal{L}_{\mathrm{Reg}} = -\frac{\tilde{\boldsymbol{y}}_{\mathrm{p}} \cdot \tilde{\boldsymbol{y}}}{||\tilde{\boldsymbol{y}}_{\mathrm{p}}||_2 \cdot ||\tilde{\boldsymbol{y}}||_2}, \tag{7}$$

where $\tilde{\boldsymbol{y}}_{\mathrm{p}}$ is a state predicted only by the physics model $\dot{x} = f_{\mathrm{p}}(x, u; \theta_{\mathrm{p}})$.

3.4. Fine-tuning in target domains

We assume that a dataset in the target domain $\mathcal{D}_\mathrm{t} \coloneqq \{(x_0, \boldsymbol{u}, \boldsymbol{y})\}$ is available after the concept shift:

$$p_\mathrm{s}(\boldsymbol{y} \mid x_0, \boldsymbol{u}) \neq p_\mathrm{t}(\boldsymbol{y} \mid x_0, \boldsymbol{u}),\ p_\mathrm{s}(x_0, \boldsymbol{u}) = p_\mathrm{t}(x_0, \boldsymbol{u}). \tag{8}$$

Our objective is to acquire an accurate model to describe the behavior in the target domain with few-shot learning, supposing $|\mathcal{D}_\mathrm{t}| \ll |\mathcal{D}_\mathrm{s}|$. To this end, we utilize the model pre-trained in the source domain and adapt it to the target domain with fine-tuning. We show the training algorithm in Algorithm 1, which only differs from the training in the source domain, in that pre-trained parameters θ_p^* and θ_d^* are used for the initial values of θ_p and θ_d.

Algorithm 1 Domain adaptation of hybrid neural ODEs

Require: dataset $\mathcal{D}_\mathrm{t}$ and pre-trained parameters $\theta_\mathrm{p}^*, \theta_\mathrm{d}^*$.
1: **Initialize**: $\theta \coloneqq [\theta_\mathrm{p}, \theta_\mathrm{d}] \leftarrow [\theta_\mathrm{p}^*, \theta_\mathrm{d}^*]$.
2: **for** *Epoch* $= 1$ to N **do**
3: **for all** $(x_0, \boldsymbol{u}, \boldsymbol{y}) \in \mathcal{D}_\mathrm{t}$ **do**
4: Propagate $\dot{x} = f_\mathrm{p}(x, u; \theta_\mathrm{p}) + f_\mathrm{d}(x, u; \theta_\mathrm{d})$ from the initial state x_0, and calculate the trajectory $\tilde{\boldsymbol{y}} = [x_0, \tilde{x}_1, \ldots, \tilde{x}_T]$.
5: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta(||\boldsymbol{y} - \tilde{\boldsymbol{y}}||_2/T + \lambda \cdot \mathcal{L}_\mathrm{Reg})$.
6: **end for**
7: **end for**

4. Experiments

4.1. Problem settings and baseline

We conduct experiments to evaluate the domain adaptation capabilities of the hybrid models with fine-tuning. In this study, we take multicopters as an example of dynamical systems with control inputs and consider two types of concept shifts, parameter shifts and external forces. In the parameter shifts, the physics parameters in f_p vary, while dynamics not included in f_p change when the external forces act on the vehicle.

We compare the fine-tuned hybrid neural ODEs with baseline models that have no physics component, i.e., the standard neural ODE models. The baselines are trained in the same way as the hybrid models: pre-trained with the source domain dataset $\mathcal{D}_\mathrm{s}$ and fined-tuned with the target domain set $\mathcal{D}_\mathrm{t}$ subsequently. After the training, the models are evaluated with two metrics: the prediction error in the target domain after the domain adaptation, $\mathcal{L}_\mathrm{MSE}^\mathrm{a}$; and the performance improvement ratio

$$\rho_\mathrm{s} = \frac{\mathcal{L}_\mathrm{MSE}^\mathrm{s} - \mathcal{L}_\mathrm{MSE}^\mathrm{a}}{\mathcal{L}_\mathrm{MSE}^\mathrm{s}}, \tag{9}$$

where $\mathcal{L}_\mathrm{MSE}^\mathrm{s}$ is the prediction errors trained only with the source domain data $\mathcal{D}_\mathrm{s}$. ρ_s represents how much the prediction error was reduced by fine-tuning.

4.2. Multicopter dynamics

Let $\boldsymbol{p}$ be the position, $\boldsymbol{q}$ be the quaternion representing the attitude, $\boldsymbol{v}$ be the velocity, and $\boldsymbol{\omega}$ be the angular velocity of a multicopter. The equation of motion of a multicopter can be written as follows (Torrente et al., 2021; Salzmann et al., 2023):

$$\dot{\boldsymbol{x}} = \frac{d}{dt}\begin{bmatrix}\boldsymbol{p}\\ \boldsymbol{q}\\ \boldsymbol{v}\\ \boldsymbol{\omega}\end{bmatrix} = f_{\mathrm{p}}(\boldsymbol{x}, \boldsymbol{u}; \theta_{\mathrm{p}}) = \begin{bmatrix}\boldsymbol{v}\\ \frac{1}{2}\boldsymbol{q}\otimes\boldsymbol{\omega}\\ \frac{1}{m}\boldsymbol{q}\otimes\boldsymbol{T}\otimes\bar{\boldsymbol{q}} + \boldsymbol{g}\\ J^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega}\times J\boldsymbol{\omega})\end{bmatrix}, \quad \boldsymbol{T} = \begin{bmatrix}0, 0, -\sum_{i=1}^{n} k\Omega_i^2\end{bmatrix}^T, \tag{10}$$

where $\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\omega}]^T$ is the state, $\boldsymbol{u}$ is the rotation of propellers, m is the mass, $\boldsymbol{g}$ is the gravity vector, $J = \mathrm{diag}(I_x, I_y, I_z)$ is the inertia matrix, k is the rotor coefficient, n is the number of rotors, and $\boldsymbol{T}, \boldsymbol{\tau}$ are the thrust and torque vectors in the body frame, respectively. The notation $\otimes$ stands for the quaternions multiplication, and $\bar{q}$ is the conjugate of the quaternion q. In this experiment, we set $\theta_{\mathrm{p}} = [I_x, I_y, k]$ as the unidentified parameters to train. We incorporate Eq. (10) as the physics component in the hybrid model in Eq. (4).

4.3. Source domain

Situations In the source domain, we consider aerodynamic drag forces and ground effects as modeling errors that the physics model f_{p} fails to represent, making accurate system identification using f_{p} alone challenging. We suppose drag force $\boldsymbol{D}$ is proportional to the velocity, i.e., $\boldsymbol{D} = -C_D\boldsymbol{v}$ with a drag coefficient C_D. Ground effects are phenomena that increase the thrusts of the rotors when the multicopters are flying at a low altitude. We adopt the following ground effect model explained in Throneberry et al. (2021) for the experiment in this section:

$$\frac{T}{T_\infty} = \frac{1}{1 - \left(\frac{R}{4h}\right)^2}, \tag{11}$$

where T and T_∞ are the thrust with and without the ground effect, respectively, R is the rotor radius, and h is the altitude. We use the multicopter dynamics with the drag and ground effect as the data-generating process in the source domain. The physics model in (10) is not aware of these terms, which motivates the use of the hybrid model in (4).

Datasets and training We generate the source domain dataset $\mathcal{D}_{\mathrm{s}}$ with the Pegasus Simulator (Jacinto et al., 2024). The numbers of trajectories are 10000 for the training and 1000 for the validation. In each trajectory, the prediction horizon h is 10 and the time-step Δt is 0.01. We train the hybrid model of multicopters $\dot{x} = f_{\mathrm{p}}(x, u) + f_{\mathrm{d}}(x, u)$ with this dataset $\mathcal{D}_{\mathrm{s}}$ to obtain the pre-trained model, following the training procedure described in Sec. 3.3. We conduct 10 trials per each condition, changing random seed values. The details of implementation are summarized in Appendix A.

4.4. Domain adaptation to target domain (1): Parameter shift

Situations In this section, we consider a situation where the parameters included in the physics model θ_{p} are changed, which is one of the possible concept shift problems for robotic systems. Table 1 shows the list of the shifted parameters. Our aim is to modify the pre-trained model by fine-tuning so that it captures the effect of the parameter shift.

Table 1: Shifted parameters of the multicopter.

Parameter	Source domain $\mathcal{D}_\mathrm{s}$	Target domain $\mathcal{D}_\mathrm{t}$
I_x [kg $\cdot$ m^2]	0.02912	0.04000
I_y [kg $\cdot$ m^2]	0.02912	0.02000

Datasets and training We implement the parameter shift on the same simulator as that used for making the source domain dataset. We generate the target dataset $\mathcal{D}_\mathrm{t}$ with the random flight in the simulator. The numbers of the training trajectories are {100, 300, 500, 700, 1000}, fewer than 10000 trajectories in the source domain. For example, 100 trajectories correspond to 10 seconds of flight data in total. The validation and test sets have 1000 trajectories each. A trajectory consists of 10 steps of prediction with the time-step $\Delta t = 0.01$ for the training and validation, and $\Delta t = 0.1$ for the test sets.

The models are fine-tuned following Algorithm 1. We conduct 10 trials per each condition, changing random seed values. After the training, the models were evaluated with the test datasets. To compare the result, we also test the source-only models (which is trained in the source domain data, and not fine-tuned) and target-only models (which is trained only with the target data without pre-training).

Results Fig. 2 showcases the predictions for one of the test trajectories with varying target domain data sizes. We see the prediction by the hybrid model are consistently closer to the ground truth than those by the no-physics baseline, which implies the effectiveness of our hybrid neural ODE framework. Fig. 3 quantitatively compares the hybrid approach and baseline in terms of two metrics $\mathcal{L}_\mathrm{MSE}$ and ρ_s. In each model, $\mathcal{L}_\mathrm{MSE}$ decreases as the number of target training trajectories increases, and by fine-tuning, the prediction errors are generally reduced from the source-only models. Although the prediction errors of target-only and fine-tuned models are almost equal in the hybrid model trained with 500 or more trajectories due to their high learning capabilities, the hybrid model shows better prediction results than the target-only models especially when the dataset size is small.

Comparing the fine-tuned hybrid model and no-physics baseline, the hybrid model results in much less prediction error $\mathcal{L}_\mathrm{MSE}$ and higher improvement ratio ρ_s than the no-physics counterparts. Especially, while the no-physics model shows a negative improvement ratio and fails in domain adaptation when trained only with 100 trajectories, the hybrid one succeeds in fine-tuning. Moreover, the standard deviation of the hybrid model's performance is notably smaller than that of the baseline, which suggests that the hybrid model is more robust against the effects of randomness.

4.5. Domain adaptation to target domain (2): Additional external forces

Situations In this section, we experiment with situations where external periodic forces act on the multicopter, in addition to the drags and ground effects. We apply $[0, -4 + \sin(2/3)\pi t, 0]^T$ [N] force to the center of gravity in the inertial coordinate, which simulates the wind disturbances during the flight. Such additional force should be captured by the fine-tuned neural network f_d.

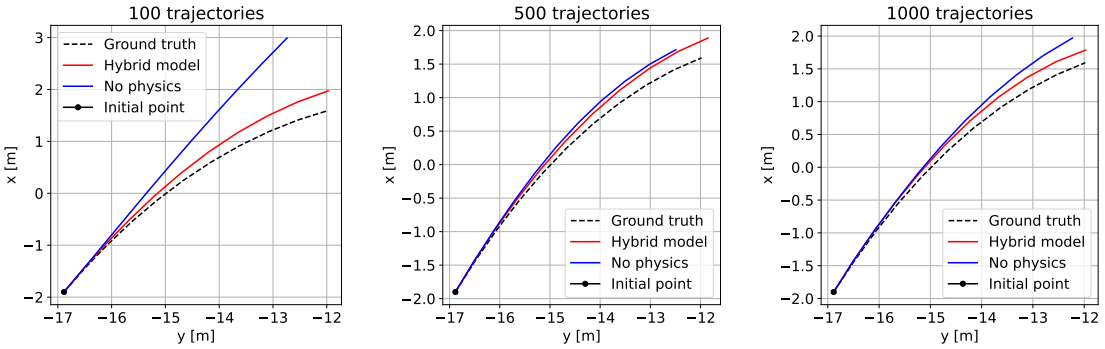

Figure 2: Examples of predicted trajectories in the target domain after the parameter shift with the different numbers of target training trajectories. Each trajectory shows the multicopter motion in the horizontal plane starting from the initial point.

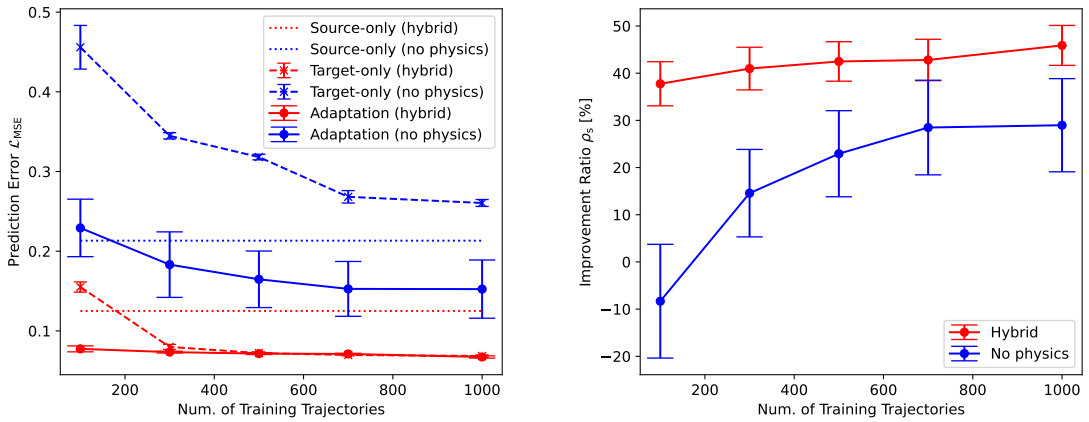

Figure 3: Comparison of the hybrid model and the no-physics baseline in the parameter shift experiment, in terms of (*left*) prediction error $\mathcal{L}_{\mathrm{MSE}}$ for the test data in the target domain; and (*right*) performance improvement ratio by fine-tuning, ρ_{s}. The error bars represent the standard deviation over 10 random trials.

Datasets and training The conditions of the datasets and training are the same as the experiment with the parameter shift. We implement the parameter shift on the same simulator as that used for making the source domain dataset. We generate the target dataset $\mathcal{D}_\mathrm{t}$ with the random flight in the simulator. The numbers of the training trajectories are $\{100, 300, 500, 700, 1000\}$, fewer than 10000 trajectories in the source domain. The validation and test sets have 1000 trajectories each. The other settings as well are the same as the previous experiment.

Results The trend is similar to the results of the parameter shift experiments. Fig. 4 shows predicted trajectories. Fig. 5 is the quantitative comparison between the hybrid model and baseline in terms of two metrics $\mathcal{L}_\mathrm{MSE}$ and ρ_s. Again, the hybrid model shows much smaller prediction error $\mathcal{L}_\mathrm{MSE}$ and higher improvement ratio ρ_s than the no-physics counterpart does. Particularly, while the results of the no-physics model trained only with 100 trajectories show divergence, probably because the dataset size was too small to fine-tune the pre-trained model, the hybrid model successfully decreased the prediction error. Besides, the standard deviation of hybrid model is consistently small compared to the baseline.

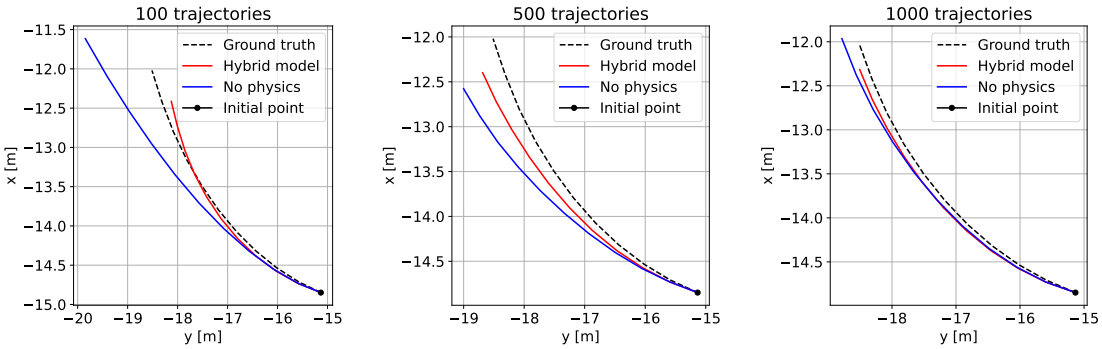

Figure 4: Examples of predicted trajectories in the target domain after applying the external force with the different numbers of training trajectories. Each trajectory shows the multicopter motion in the horizontal plane starting from the initial point.

4.6. Discussions

In the two experiments, the hybrid neural ODEs consistently outperformed the standard neural ODE (no-physics) baselines in terms of both metrics, $\mathcal{L}_\mathrm{MSE}$ and ρ_s. It indicates that incorporating physics-based models into neural ODEs not only improves the prediction accuracy in general but also enhances the domain adaptation capabilities after concept shifts in either physics or data-driven components. In addition, the fine-tuned hybrid models trained only with 100 target domain trajectories ($\approx$ 10 seconds of flight) achieved almost equivalent prediction performance to the models trained with 1000 target domain data ($\approx$ 100 seconds of flight). It shows that the utilization of pre-trained hybrid models reduces the quantity of data required in the target domain and enables efficient learning to deal with data scarcity.

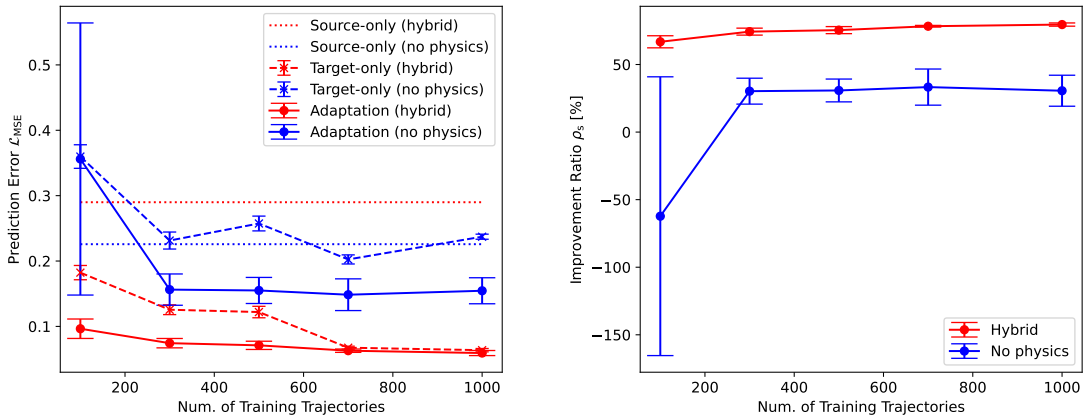

Figure 5: Comparison of the hybrid model and the no-physics baseline in the experiment with the external force, in terms of (*left*) prediction error $\mathcal{L}_{\mathrm{MSE}}$ for the test data in the target domain; and (*right*) performance improvement ratio by fine-tuning, ρ_{s}.

We would attribute the superior performance of hybrid models to two main factors. First, by incorporating physics-based structures and pre-training, the dominant dynamics are already captured by the physics components. As a result, the role of the neural network is simplified compared to models without physics, making the parameter optimization during fine-tuning more tractable. Second, the integration of physics components introduces an inductive bias that constrains model flexibility. This acts as a form of regularization, helping to prevent overfitting especially when the amount of available target data is limited. Elaborating on the detailed mechanism of fine-tuning with hybrid models would be an interesting direction of research.

4.7. Comparison with related methods

In this section, we conduct a further experiment with Expert Augmentation (Wehenkel et al., 2023), a notable hybrid modeling method that we consider most relevant to our approach, as it also addresses distributional shifts using physics-based components. The key idea of Expert Augmentation is to train an encoder to predict physics-grounded parameters using augmented data. Following Wehenkel et al. (2023), we perform data augmentation by randomly drawing θ_{p} from a plausible range and then trained an encoder with the augmented data. The problem setting was slightly altered so that the encoder takes past states of length 10 as input. Unlike Wehenkel et al. (2023), the encoder here newly appears, as we do not have (or need) an encoder originally. We then evaluate the performance of the Expert Augmentation model with the new encoder on the target domain. In addition, we compare the results with a physics-only model, which we pre-train with the source data and then fine-tune with target data in the same manner as our hybrid models.

The experiments are carried out under two problem settings corresponding to those in Sec. 4.4 and 4.5. The conditions are identical to these experiments above, except that the time step Δt is fixed at 0.01 in the test data (aligned with the training data) for simplicity.

In both cases, our hybrid model is fine-tuned using only 100 target-domain data points (the most challenging case with the least available target data), and we conduct 10 trials with different random seeds.

Table 2: Test loss (MSE) comparison with Expert Augmentation and physics-only models in Experiment 1 (parameter shifts) and Experiment 2 (external forces). Each value represents an average or minimum of 10 trials.

	Test loss (Experiment 1)	Test loss (Experiment 2)
Expert Augmentation (Min.)	3.68×10^{-3}	4.02×10^{-3}
Physics-only model (Min.)	3.76×10^{-3}	4.53×10^{-3}
Our hybrid model (Average)	$3.26 \times 10^{-3} \pm 2.66 \times 10^{-5}$	$2.64 \times 10^{-3} \pm 1.56 \times 10^{-4}$

The results are shown in Table 2. It indicates that while Expert Augmentation performs better than the physics-only model, our hybrid model still outperforms Expert Augmentation and achieves the best overall performance.

We note that the problem setting supposed by Expert Augmentation (Wehenkel et al., 2023) and ours are slightly different, which may partly explain the empirical results. In Wehenkel et al. (2023), the marginal distribution of the explanatory variables is supposed to be unchanged even under overall distribution shifts. On the other hand, in our experiments, the distribution of the explanatory variables may be different between source and target domains. This is because our explanatory variables include control signals, which depend on the environment.

5. Conclusion

To address the performance degradation of neural dynamical system models under domain shifts, we have focused on the fine-tuning of hybrid models that incorporate physics models into neural ODEs. Specifically, we adopted an additive formulation of the physics and data-driven components to compensate for incomplete physics models with some deviations from the real world (e.g., unidentified parameters and unmodeled forces). Since these physics structures following equations of motion are consistent by nature even after domain shifts, we investigated how these invariant features contribute to the domain adaptation capabilities. We empirically validated the hybrid neural ODEs using multicopter dynamics as a representative case of dynamical systems with control inputs. The experimental results indicated that our hybrid modeling framework contributed to improving domain adaptation capabilities after the concept shift, outperforming conventional neural ODE models. Besides, our fine-tuning strategy reduced the number of required data in the target domain, enabling efficient few-shot learning with limited label data.

Despite these promising results, several challenges to address remain for future work. Firstly, our discussions are limited to ODEs, and extending the approach to partial differential equations (PDEs) with high dimensional states is necessary to consider more complicated physics such as fluid dynamics governed by the Navier–Stokes equations. We also plan to apply the hybrid modeling framework to model-based control tasks in robotics sys-

tems subject to domain shifts, and evaluate the effects of our few-shot domain adaptation approach on improving control performance.

Acknowledgments

This work was supported by JSPS International Joint Research Program JPJSJRP20221501, JST PRESTO JPMJPR24T6, JSPS KAKENHI JP24K01110, and JST SPRING JPMJSP2108.

References

S. Arik, C.L. Li, J. Yoon, R. Sinha, A. Epshteyn, L. Le, V. Menon, S. Singh, L. Zhang, M. Nikoltchev, Y. Sonthalia, H. Nakhost, E. Kanal, and T. Pfister. Interpretable sequence learning for covid-19 forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18807–18818, 2020.

L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza. Neurobem: Hybrid aerodynamic quadrotor model. In *Robotics: Science and Systems XVII (RSS)*, 2021.

L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

A. Farahani, S. Voghoei, K. Rasheed, and H. R. Arabnia. A brief review of domain adaptation. In *Advances in Data Science and Information Engineering*, pages 877–894. Springer International Publishing, 2021.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481, 2021.

L. Hewing, J. Kabzan, and M. N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2020a.

L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020b.

M. Jacinto, J. Pinto, J. Patrikar, J. Keller, R. Cunha, S. Scherer, and A. Pascoal. Pegasus simulator: An isaac sim framework for multiple aerial vehicles simulation. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 917–922, 2024.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2017.

M. Kirchmeyer, Y. Yin, J. Dona, N. Baskiotis, A. Rakotomamonjy, and P. Gallinari. Generalizing to new physical systems via context-informed dynamics model. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 11283–11301, 2022.

J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.

R. D. Nzoyem, D. A.W. Barton, and T. Deakin. Neural context flows for meta-learning of dynamical systems. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

A. Osaka, N. Takeishi, and T. Yairi. Deterministic and stochastic hybrid modeling with regularization. In *Proceedings of 2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2025.

T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, 2023.

P. Singhal, R. Walambe, S. Ramanna, and K. Kotecha. Domain adaptation: Challenges, methods, datasets, and applications. *IEEE Access*, 11:6973–7020, 2023.

N. Takeishi and A. Kalousis. Physics-integrated variational autoencoders for robust and interpretable generative modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14809–14821, 2021.

N. Takeishi and A. Kalousis. Deep grey-box modeling with adaptive data-driven models toward trustworthy estimation of theory-driven models. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4089–4100, 2023.

G. Throneberry, C.M. Hocut, and A. Abdelkefi. Multi-rotor wake propagation and flow development modeling: A review. *Progress in Aerospace Sciences*, 127:100762, 2021.

G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza. Data-driven mpc for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):3769–3776, 2021.

Y. Verma, M. Heinonen, and V. Garg. Climode: Climate and weather forecasting with physics-informed neural odes. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

R. Wang, D. Maddix, C. Faloutsos, Y. Wang, and R. Yu. Bridging physics-based and data-driven modeling for learning dynamical systems. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control (L4DC)*, pages 385–398, 2021.

R. Wang, R. Walters, and R. Yu. Meta-learning dynamics forecasting using task inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21640–21653, 2022.

A. Wehenkel, J. Behrmann, H. Hsu, G. Sapiro, G. Louppe, and J. H. Jacobsen. Robust hybrid learning with expert augmentation. *Transactions on Machine Learning Research*, 2023.

Y. Yin, V. Le Guen, J. Dona, E. de Bézenac, I. Ayed, N. Thome, and P. Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021.

Appendix A. Implementation details

A.1. Neural network architectures

We employed the same neural network architectures for the hybrid model and the no-physics baseline. Each neural network f_{d} has one hidden layer with 64 nodes and the layers are fully connected. The activation function is ReLU and batch normalization is introduced.

A.2. Optimization in the pre-training

For the hybrid model, the learning rate is set to be 1×10^{-5} for θ_{p} and 1×10^{-4} for θ_{d}. Training is performed for at least 100 epochs and continues until the validation loss does not improve for 20 consecutive epochs. For the no-physics baseline, the learning rate for θ_{d} is 1×10^{-4}, the minium epochs are 300, and training is continued until the validation loss fails to improve for 50 consecutive epochs. PyTorch is adopted for the implementation and Adam (Kingma and Ba, 2017) is chosen for the optimizer. We use NVIDIA GeForce RTX 3090 GPU for computation.

A.3. Optimization in the fine-tuning

The conditions are the same in the two fine-tuning experiments: the parameter shifts and the external forces. For the hybrid model fine-tuned with {100, 300, 500, 700, 1000} target data, the learning rate is set to be 1×10^{-5} for θ_{p} and $\{1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-3}\}$ for θ_{d}, respectively. The training is conducted for a minimum of {100, 300, 300, 300, 100} epochs and continued until the validation loss fails to improve for {20, 50, 50, 50, 20} consecutive epochs. For the no-physics baseline, the learning rate for θ_{d} is $\{1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$, the minium epochs are 250, and training is continued until the validation loss fails to improve for 50 consecutive epochs in all the conditions. PyTorch is adopted for the implementation and Adam (Kingma and Ba, 2017) is chosen for the optimizer. We use NVIDIA GeForce RTX 3090 GPU for computation.

Appendix B. Data generation

The Pegasus Simulator (Jacinto et al., 2024) is a simulation framework for multicopters working on NVIDIA Isaac Sim and offers graphics with high resolution as shown in Fig. 6. As a multicopter model, we use the Iris Quadrotor, which is incorporated in the Pegasus Simulator. Fig. 7 shows an example of flight trajectories generated with the simulator. We generated the datasets by the manual flight in the simulator.

Figure 6: Simulation environment in the Pegasus Simulator (Jacinto et al., 2024).

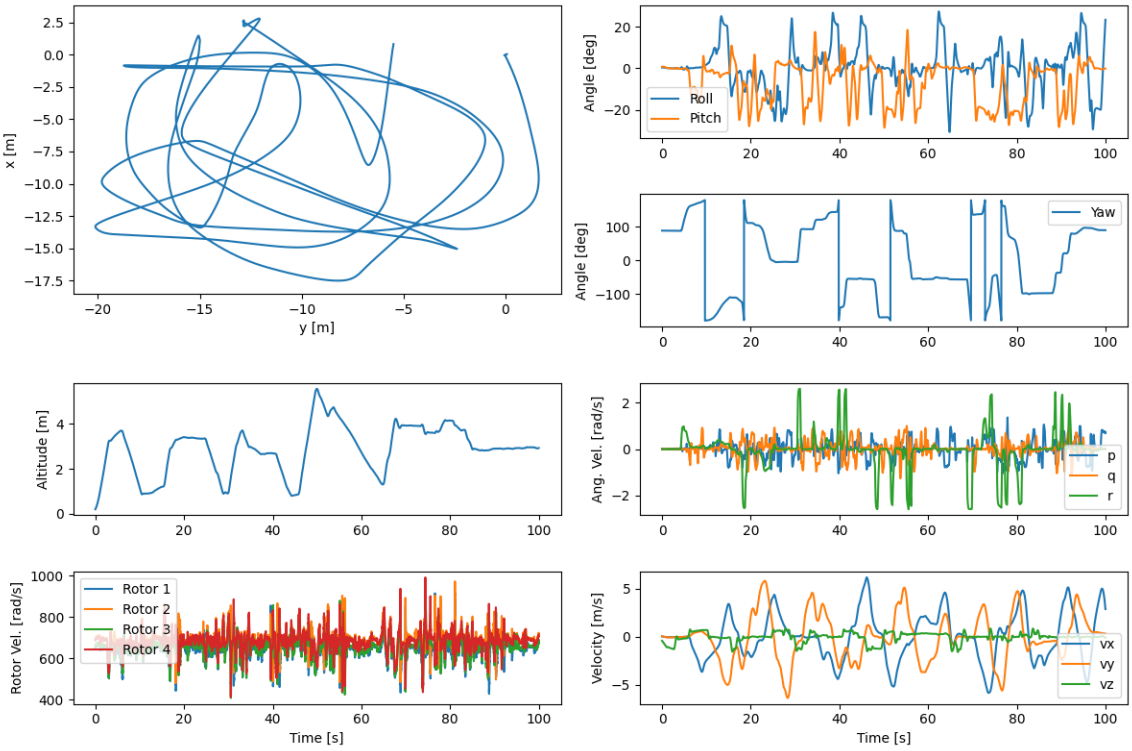

Figure 7: An example of simulated data.