Deep Grey-Box Modeling With Adaptive Data-Driven Models Toward Trustworthy Estimation of Theory-Driven Models

Naoya Takeishi and **Alexandros Kalousis**
University of Applied Sciences and Arts Western Switzerland (HES-SO), Geneva, Switzerland

Abstract

The combination of deep neural nets and theory-driven models (*deep grey-box models*) can be advantageous due to the inherent robustness and interpretability of the theory-driven part. Deep grey-box models are usually learned with a regularized risk minimization to prevent a theory-driven part from being overwritten and ignored by a deep neural net. However, an estimation of the theory-driven part obtained by uncritically optimizing a regularizer can hardly be trustworthy if we are not sure which regularizer is suitable for the given data, which may affect the interpretability. Toward a trustworthy estimation of the theory-driven part, we should analyze the behavior of regularizers to compare different candidates and to justify a specific choice. In this paper, we present a framework that allows us to empirically analyze the behavior of a regularizer with a slight change in the architecture of the neural net and the training objective.

1 INTRODUCTION

Grey-box modeling generally refers to combining theory-driven structures and data-driven components (e.g., Sohlberg and Jacobsen, 2008). We are particularly interested in the combination of theory-driven models such as mathematical descriptions of physical phenomena and data-driven models. For example, for regression from x to y, the models can be like (but not limited to)

$$y = f_{\mathrm{T}}(x; \theta_{\mathrm{T}}) + f_{\mathrm{D}}(x; \theta_D) + e,$$

where f_{T} and f_{D} denote theory- and data-driven models parameterized by θ_{T} and θ_{D}, respectively, and e is noise.

Proceedings of the 26[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

We discuss how we should formulate the estimation of grey-box models with an emphasis on their interpretation. Because grey-box models partly rest on a theoretical backbone, they can be reasonably interpretable and thus useful for gleaning insights from data for which our theory is essentially incomplete. However, interpretability is not a free lunch, and we should pay attention to the estimation process in order to ensure trustworthy interpretation. The estimation could be directly analyzed for simple models, but it is usually not feasible for general nonlinear models.

We are interested in cases where the data-driven model, f_{D}, is a deep neural network (Mehta et al., 2021; Qian et al., 2021; Takeishi and Kalousis, 2021; Wehenkel et al., 2023; Yin et al., 2021, see also Section 4), where care must be taken that the theory-based model, f_{T}, is not overwritten and ignored by f_{D} due to the expressiveness of the latter. Such *deep grey-box models* are typically learned by regularized risk minimization such as $\mathrm{minimize}_{\theta_{\mathrm{T}}, \theta_{\mathrm{D}}} \mathcal{L} + \lambda \mathcal{R}$, where $\mathcal{L}$ is a prediction loss, and $\mathcal{R}$ is the sum of regularizers including those to prevent f_{T} from being ignored.

In fact, $\mathcal{R}$ completely dictates the choice of θ_{T} when the empirical $\mathcal{L}$ is minimized to the same extent for different θ_{T} due to the expressiveness of f_{D}. Knowing the nature of $\mathcal{R}$ means knowing how θ_{T} is chosen; it is an important clue for the trustworthy interpretation of the model because θ_{T} is at the core of the interpretable part of the model. Moreover, if we have multiple candidates of $\mathcal{R}$, knowing each $\mathcal{R}$'s property is important for choosing one or a combination of them. However, it is usually difficult to describe the nature of $\mathcal{R}$ *a priori* because it depends on the outputs of f_{D} and f_{T}; the former is a deep neural net, and the latter can also be a nonlinear function. Therefore, we propose a method to empirically analyze the nature of $\mathcal{R}$.

Our idea is simple: We withhold the estimation of θ_{T}, at least in the first attempt at data analysis. We learn f_{D} adaptively with realizations of θ_{T}, essentially marginalizing out θ_{T}. Such a slight change in the formulation allows us to analyze $\mathcal{R}$ empirically by examining its landscape without the need for re-training the model. In the rest of the paper, we take up the above argument about the trustworthiness of the estimation, formulate the idea, and conduct an empirical investigation of its effectiveness.

A useful by-product of the formulation is that the minimization of $\mathcal{L}$ and $\mathcal{R}$ can now be decoupled. This allows us to use different optimizers for the two objectives, as well as to use unlabeled test data to optimize $\mathcal{R}$. Moreover, we have found that such a decoupled process makes the optimization much less sensitive to the hyperparameter, λ.

2 PRELIMINARY

2.1 Definition

We define *deep grey-box models* as compositions of theory-driven models and data-driven models, with the latter being deep neural networks. For the sake of discussion, we suppose regression problems where $y \in \mathcal{Y}$ is to be predicted from $x \in \mathcal{X}$, though the extension to other problems is straightforward. We denote such a model in general by

$$y = \mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; x), \tag{1}$$

where $\mathcal{C}$ is a functional that takes the two types of functions and an input variable as arguments. The two functions, f_{T} and f_{D}, are a theory-driven model and a deep neural network, with unknown parameters $\theta_{\mathrm{T}} \in \Theta_{\mathrm{T}}$ and $\theta_{\mathrm{D}} \in \Theta_{\mathrm{D}}$, respectively. We may write $f_{\mathrm{T}}(x; \theta_T)$ to manifest f_{T}'s dependency on θ_{T} or write simply $f_{\mathrm{T}}(x)$ though it still depends on θ_{T} (and analogously for f_{D} and θ_{D}). We usually expect $\dim \theta_{\mathrm{T}} \ll \dim \theta_{\mathrm{D}}$. The functional, $\mathcal{C}$, evaluates f_{T} and f_{D} with a given $(\theta_{\mathrm{T}}, \theta_D, x)$ and then mixes up their outputs to give the final output of the model.

We try to keep the generality of $\mathcal{C}$; it may include general function compositions and their arbitrary transformations:

$$\mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; x) = \mathrm{SomeTransformation}[f_{\mathrm{D}}(f_{\mathrm{T}}(x), x)].$$

Meanwhile, one of the most prevailing forms of $\mathcal{C}$ in the literature is the additive grey-box ODEs like:

$$\mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; x) = \mathrm{ODESolve}[\dot{s} = f_{\mathrm{T}}(s) + f_{\mathrm{D}}(s) \mid s_0 = x],$$

where s is the state variable of the dynamics, $\dot{s}$ is its time derivative, s_0 is the initial condition, and ODESolve denotes an operation that numerically solves initial value problems. Such grey-box (ordinary or partial) differential equations have been studied by researchers such as Mehta et al. (2021), Qian et al. (2021), Sasaki et al. (2019), Takeishi and Kalousis (2021), and Yin et al. (2021).

We are particularly interested in cases where $\mathcal{C}$ inherits the expressive power of f_{D} as a function approximator. This is *not* the case, for example, when $\mathcal{C}$ only contains compositions such as $f_{\mathrm{T}}(f_{\mathrm{D}}(x), x)$, that is, f_{T} is "outside" f_{D} (e.g., Arık et al., 2020; Raissi et al., 2019; Schnell et al., 2022). Estimation of such models is generally less problematic in the sense that f_{T} cannot be ignored by construction. In contrast, we address cases where f_{T} is "inside" f_{D}, for which we should be careful so that f_{T} is not overwritten and ignored by f_{D}. We put the following assumptions on the model:

Assumption 1. $f_{\mathrm{D}} \colon \mathcal{X} \to \mathcal{Y}$ is a universal function approximator; for any $\epsilon > 0$ and a continuous function $g \colon \mathcal{X} \to \mathcal{Y}$, there exists $\theta_{\mathrm{D}} \in \Theta_{\mathrm{D}}$ satisfying $\sup_{x \in S_X} \| f_{\mathrm{D}}(x; \theta_{\mathrm{D}}) - g(x) \| < \epsilon$, where $S_X \subset \mathcal{X}$ is some compact set.

Assumption 2. $\mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; \cdot) \colon \mathcal{X} \to \mathcal{Y}$ is also a universal function approximator; that is, for any $\epsilon > 0$, $\theta_{\mathrm{T}} \in \Theta_{\mathrm{T}}$, and a continuous function $g' \colon \mathcal{X} \to \mathcal{Y}$, there exists $\theta_{\mathrm{D}} \in \Theta_{\mathrm{D}}$ satisfying $\sup_{x \in S_X} \| \mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; x) - g'(x) \| < \epsilon$.

Remark 1. We assume the universal approximation property just to construct the discussion rigorously. Even without the universal approximation property, as long as f_{D} and $\mathcal{C}$ are much more expressive than f_{T}, discussions below would approximately hold in practice.

Assumption 3. f_{T} and f_{D} are Lipschitz continuous with regard to (x, θ_{T}) and (x, θ_{D}), respectively.

2.2 Why Grey-box?

Deep grey-box models are powerful function approximators with a certain level of inherent interpretability owing to f_T, a human-understandable model with a theory as a backbone. A typical use case would be to estimate a grey-box model on data on which our theory is essentially incomplete and inspect the estimated model to glimpse insights, e.g., when the incomplete theory is correct or not, how the missing part approximated by f_{D} behaves, and so on.

Deep grey-box models can also be advantageous in generalization capability and robustness to extrapolation, as reported empirically (Qian et al., 2021; Takeishi and Kalousis, 2021; Wehenkel et al., 2023; Yin et al., 2021). Although rigorous analysis is yet to be done for many cases, it is natural to expect such improvements because the presence of f_{T} in the model would reduce the sample complexity of the learning problem, and f_{T} is supposed to work well in the out-of-data regime (or rather, it is a requirement for a model to be theory-driven). In this paper, we do not touch on such performance aspects of deep grey-box models given the previous studies, so the comparison to non-grey-box models is out of the paper's scope.

2.3 Empirical Risk Minimization Cannot Select θ_T

There is a natural consequence of deep grey-box modeling; the theory-driven model's parameter, θ_{T}, cannot be chosen solely by minimizing an empirical risk of prediction. For example, suppose we learn $\mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; x) = f_{\mathrm{T}}(x) + f_{\mathrm{D}}(x)$ by minimizing the mean squared error, $\mathcal{L} = \| y - (f_{\mathrm{T}}(x) + f_{\mathrm{D}}(x)) \|_2^2$. The empirical risk can be minimized to a similar extent for *any* $\theta_{\mathrm{T}} \in \Theta_{\mathrm{T}}$ because f_{D}, a deep neural net, can, it alone, approximate any function on the training set (as assumed in Assumption 1) and thus also the function $y - f_{\mathrm{T}}(x)$. We formally state this fact as follows:

Proposition 1. *Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set. Let $\mathcal{L}(\theta_\mathrm{T}, \theta_\mathrm{D}; x, y)$ be a Lipschitz continuous loss function between the prediction (i.e., the value of $\mathcal{C}(f_\mathrm{T}, f_\mathrm{D}; x)$) and the target (i.e., y). Let $\mathcal{L}_S(\theta_\mathrm{T}, \theta_\mathrm{D}) = \sum_{(x,y)\in S} \mathcal{L}(\theta_\mathrm{T}, \theta_\mathrm{D}; x, y)$ be the empirical risk on the training set. Suppose that Assumptions 1–3 hold. Then, for any $\epsilon' > 0$, $\theta_\mathrm{D} \in \Theta_\mathrm{D}$, and $\theta_\mathrm{T}, \theta'_\mathrm{T} \in \Theta_\mathrm{T}$, there exists $\theta'_\mathrm{D} \in \Theta_\mathrm{D}$ that satisfies*

$$|\mathcal{L}_S(\theta_\mathrm{T}, \theta_\mathrm{D}) - \mathcal{L}_S(\theta'_\mathrm{T}, \theta'_\mathrm{D})| < \epsilon'. \tag{2}$$

Proof. From the assumptions, for any $\epsilon > 0$, $\theta_\mathrm{D} \in \Theta_D$, and $\theta_\mathrm{T}, \theta'_\mathrm{T} \in \Theta_\mathrm{T}$, there exists $\theta'_\mathrm{D} \in \Theta_D$ that satisfies $\sup_{x\in\{x_1,\ldots,x_n\}} \|\mathcal{C}(f_\mathrm{T}, f_\mathrm{D}; x) - \mathcal{C}(f'_\mathrm{T}, f'_\mathrm{D}; x)\| < \epsilon$, where f'_i is parameterized by θ'_i for $i = \mathrm{T}, \mathrm{D}$. Since $\mathcal{L}$ is Lipschitz continuous, $\sup_{x,y\in S} |\mathcal{L}(\theta_\mathrm{T}, \theta_\mathrm{D}; x, y) - \mathcal{L}(\theta'_\mathrm{T}, \theta'_\mathrm{D}; x, y)| < K\epsilon$, where K is $\mathcal{L}$'s Lipschitz constant. Therefore, with $\epsilon' \coloneqq |S|K\epsilon$, $|\mathcal{L}_S(\theta_\mathrm{T}, \theta_\mathrm{D}) - \mathcal{L}_S(\theta'_\mathrm{T}, \theta'_\mathrm{D})| < \epsilon'$. □

2.4 Regularized Risk Minimization

Proposition 1 states that any $\theta_\mathrm{T} \in \Theta_\mathrm{T}$ can be equally likely solely under the empirical risk, $\mathcal{L}_S$. It necessitates regularizing the problem; we should optimize $\mathcal{L}_S + \lambda\mathcal{R}$ instead, where $\lambda \geq 0$ is a regularization hyperparameter, and $\mathcal{R}$ is some regularizer that should reflect our inductive biases on how we should combine the theory- and data-driven models. Let us, for example, consider the linear combination case, $\mathcal{C}(f_\mathrm{T}, f_\mathrm{D}; x) = f_\mathrm{T}(x) + f_\mathrm{D}(x)$. One of the common ways of thinking is that f_T should as accurately explain the x–y relation as possible, and f_D should have the least possible effect. This idea can be operationalized by defining $\mathcal{R} = \|f_\mathrm{D}\|$, where the norm is the function's norm. Though such an $\mathcal{R}$ has been a popular choice, it is not the only possibility. For example, when one wants the two models' output to be uncorrelated, one can use $\mathcal{R} = |\langle f_\mathrm{T}(x), f_\mathrm{D}(x)\rangle|$.[1]

We assume that $\mathcal{R}$ depends only on x. It is natural because the role of $\mathcal{R}$ is not to fit the x–y relation. We will recall this assumption, if necessary, by writing $\mathcal{R}_{S_X}$, where $S_X = \{x_1, \ldots, x_n\}$ is the extract of xs from S. We do not suppose any more specifications of $\mathcal{R}$ than this assumption.

Typically, the regularized learning problem can be formulated as inductive learning or transductive learning.

Inductive Learning The simplest formulation is to minimize $\mathcal{L} + \lambda\mathcal{R}$ with both terms evaluated only on training data, that is:

$$\theta^*_\mathrm{T}, \theta^*_\mathrm{D} \in \arg\min_{\theta_\mathrm{T}, \theta_\mathrm{D}} \mathcal{L}_S(\theta_\mathrm{T}, \theta_\mathrm{D}) + \lambda\mathcal{R}_{S_X}(\theta_\mathrm{T}, \theta_\mathrm{D}). \tag{3}$$

In this formulation, not only $\mathcal{L}$ but also $\mathcal{R}$ suffers a generalization gap. The hyperparameter, λ, needs to be tuned somehow.

[1] Suggesting specific $\mathcal{R}$ for each application or in general is out of the scope of this paper; on contrary, our proposal in Section 3 is for cases where we cannot specify $\mathcal{R}$ *a priori*.

Transductive Learning Since we assume that $\mathcal{R}$ only depends on x, *transductive learning* (Gammerman et al., 1998) can also be considered. Let S'_X be some set of x that may include S_X as a subset. The idea is to minimize the unsupervised part of the objective, not on the training data S_X but rather on S'_X;

$$\theta^*_\mathrm{T}, \theta^*_\mathrm{D} \in \arg\min_{\theta_\mathrm{T}, \theta_\mathrm{D}} \mathcal{L}_S(\theta_\mathrm{T}, \theta_\mathrm{D}) + \lambda\mathcal{R}_{S'_X}(\theta_\mathrm{T}, \theta_\mathrm{D}). \tag{4}$$

The generalization gap disappears for $\mathcal{R}$ when (a subset of) S'_X is the test set, but still λ needs to be tuned.

3 TOWARD TRUSTWORTHY ESTIMATION

3.1 Challenges of Deep Grey-box Model Estimation

Deep grey-box models have been studied mainly in terms of empirical generalization and extrapolation capability (Mehta et al., 2021; Qian et al., 2021; Takeishi and Kalousis, 2021; Wehenkel et al., 2023; Yin et al., 2021). However, when the model's interpretation is concerned, the prediction performance does not speak a lot; there can be multiple parameter values that perform similarly (cf. *Rashomon* effect (Breiman, 2001)), from which we cannot choose ones to interpret. The solution of the optimization in Eq. (3) or (4) tells us nothing about the analytical property of $\mathcal{R}$, so we can hardly understand the full picture of how the optimization selects θ_T. Instead of uncritically optimizing the regularizer, $\mathcal{R}$, we should know the property of $\mathcal{R}$ in order to gain more information to explain the choice of θ_T to be interpreted. We would contrast the situation with, for example, the estimators of linear regression models, which have been extensively analyzed and thus are trustworthy in some sense. We do not suggest analyzing our $\mathcal{R}$s analytically as it is too problem-dependent, but analyzing them at least empirically would help us make θ_T's estimation more trustworthy.

The challenge due to not knowing $\mathcal{R}$'s property stands out more when we do not know which $\mathcal{R}$ is suitable for the given data and need to compare different candidate $\mathcal{R}$s, which is often the case as we do not know the whole data-generating process. Again, the estimation via Eq. (3) or (4) would not tell much about the goodness of each $\mathcal{R}$, since different $\mathcal{R}$s could achieve similar prediction performance. This viewpoint also supports the need for analyzing $\mathcal{R}$ at least empirically for gaining information to compare different $\mathcal{R}$s.

Another challenge, yet more technical, is the choice of the regularization hyperparameter, λ. It can be tricky because, in Eq. (3) or (4), it controls two things at the same time: "which θ_T should be selected" and "how much f_D should be regularized." They are different problems if interrelated, and thus decoupling them would be beneficial.

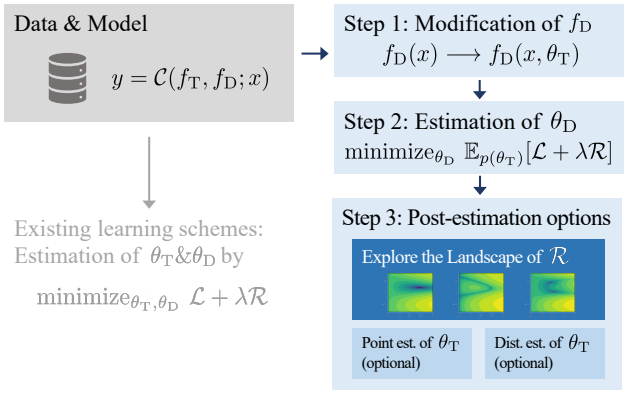

Figure 1: Overview of the proposed formulation. In contrast to the existing learning schemes where θ_T and θ_D are estimated simply by minimizing $\mathcal{L} + \lambda\mathcal{R}$ on training data (*left*), we suggest the three-step procedures toward trustworthy estimation of θ_T (*right*).

3.2 Proposed Formulation

As we argued above, analyzing $\mathcal{R}$ empirically can be a useful first step toward a trustworthy estimation of θ_T. More specifically, we aim to explore the landscape of $\mathcal{R}$, that is, to evaluate the values of $\mathcal{R}$ for different θ_Ts. The naïve way to do so is to re-run the optimization in Eq. (3) or (4) for many times with different values of θ_T fixed at each time, but it is inefficient even for a low-dimensional θ_T, and moreover, it does not allow prediction with a new value of θ_T that was not exactly tried during such many training runs. Instead, we suggest an approach to exploring $\mathcal{R}$'s landscape without the need for re-running the optimization.

At the core of our suggestion is to "marginalize out" θ_T in the training phase. To this end, first we slightly modify f_D so that it works adaptively with different values of θ_T and then minimize the objective, taking its expectation with regard to θ_T. We first estimate only θ_D while leaving θ_T undetermined; thus the idea is similar to probabilistic inference where some variables are marginalized out. The remainder of this section explains each step of the proposed formulation. Figure 1 depicts its overview.

3.2.1 Step 1: Modification of f_D

We add two specifications of f_D to be used, without much loss of generality. First, we suppose that f_D takes as arguments not only x but also θ_T (and possibly the value of $f_\mathrm{T}(x; \theta_\mathrm{T})$). Second, we define f_D's architecture such that f_D can adapt to different values of θ_T given as an argument. We can realize them with general techniques of conditional modeling; for example, by concatenating θ_T to the original input, x; or by using hypernetworks that transform θ_T to (a part of) θ_D. In the experiments in Section 5, we simply concatenate θ_T and $f_\mathrm{T}(x; \theta_\mathrm{T})$ to x and feed them to f_D either with an extended input layer or with additional channels. In what follows, we may emphasize the dependency of this new f_D on θ_T (and thus on f_T) by denoting it as $f_\mathrm{D}(f_\mathrm{T})$. Its parameters will still be denoted by θ_D.

3.2.2 Step 2: Estimation of θ_D

We then estimate θ_D by optimizing the expectation of the objective with regard to $p(\theta_\mathrm{T})$, some prior distribution of θ_T; that is,

$$\theta_\mathrm{D}^* \in \arg\min_{\theta_\mathrm{D}} \mathbb{E}_{p(\theta_\mathrm{T})}[\mathcal{L}_S(\theta_\mathrm{T}, \theta_\mathrm{D}) + \lambda\mathcal{R}_{S_X}(\theta_\mathrm{T}, \theta_\mathrm{D})], \quad (5)$$

where $\mathcal{L}_S$ and $\mathcal{R}_{S_X}$ are computed with regard to the deep grey-box model $\mathcal{C}(f_\mathrm{T}, f_\mathrm{D}(f_\mathrm{T}); x)$ with the adaptive data-driven model, $f_\mathrm{D}(f_\mathrm{T})$l, on training data S. We estimate the objective in Eq. (5) with random samples of θ_T drawn from $p(\theta_\mathrm{T})$.

The $\mathcal{R}_{S_X}$ term in Eq. (5) is present just for generality, and its choice is not important. When one has multiple candidates of regularizers, which is one of our motivating situations, it is unclear which regularizer we should use as $\mathcal{R}_{S_X}$ in Eq. (5). We suggest two options: One is to set it as the sum of all the candidate regularizers with a small value of λ. This is reasonable because it prevents the candidate regularizers from getting overly large but still does not strongly impose any one of them. Another option is to set $\lambda = 0$, with which we no longer need to set any $\mathcal{R}_{S_X}$. This is also a reasonable choice because the minimization of regularizers can be done later in the post-estimation phase as we will see.

We suppose that we have some $p(\theta_\mathrm{T})$ from which we can draw θ_Ts. It is typically available from domain knowledge concerning the theory-driven model. For example, it can be the uniform distribution on a plausible range of the parameters. It is also technically possible to let $p(\theta_\mathrm{T})$ have unknown parameters and estimate them from data with the reparametrization trick, though appropriate regularization is necessary to avoid degenerate solutions.

3.2.3 Step 3: Post-Estimation Options

The optimization in Eq. (5) leaves θ_T undetermined, while the model works with any different values of θ_T within a region having a reasonable mass of $p(\theta_\mathrm{T})$. It allows us to take the following options in the post-estimation phase.

Explore the Landscape of $\mathcal{R}$ We can compute the values of $\mathcal{R}$ for different θ_Ts, only with the cost of the forward evaluation of the functions. If θ_T is up to two-dimensional, we can directly draw the landscape of $\mathcal{R}$. If $\dim\theta_\mathrm{T} > 2$ but remains moderate (say $\lessapprox 20$), we can watch the variations of $\mathcal{R}$ by varying each pair of θ_T's elements while fixing the others at some reference values.

How should we utilize the landscape of $\mathcal{R}$? Although it is

just up to a user's policy and belief in each application, as a general practice, we suggest utilizing it for assessing the reliability of potential estimation based on the $\mathcal{R}$. For example, if an $\mathcal{R}$ does not have clear extrema along some axes of θ_{T}, it implies that those elements of θ_{T} are not quite identifiable under such an $\mathcal{R}$. It can contribute to the estimation's trustworthiness, e.g., by withholding interpretation about some parameters. Another general usage is to compare different candidates of $\mathcal{R}$. The landscapes can give an intuition about the nature of each $\mathcal{R}$, which is useful for a user to choose one (or more) out of the candidates; we will see concrete use cases later in numerical experiments.

Models with high-dimensional θ_{T} basically remain an open challenge; analyzing and visualizing a high-dimensional parameter space are very challenging in general. One of the options is to get an overview of the landscape via dimensionality reduction of the parameter space by techniques such as random projection and principal component analysis. With that being said, even if θ_{T} is high-dimensional, we can still benefit from the proposed formulation for the point estimation or the posterior inference discussed below.

Point Estimation of θ_{T} (optional) After analyzing $\mathcal{R}$s and choosing one to use, a user may want to select θ_{T} as

$$\theta_{\mathrm{T}}^* \in \arg\min_{\theta_{\mathrm{T}}} \mathcal{R}_{S'_X}(\theta_{\mathrm{T}}, \theta_{\mathrm{D}}^*). \tag{6}$$

Depending on the application, S'_X may be a singleton of a test sample, a set of test samples, the union of training and test sets, or a set of grid points on $\mathcal{X}$. After choosing a specific θ_{T}, one does not need to re-run the optimization in Eq. (5) because the estimated f_{D} works adaptively to θ_{T}.

Although the end result of Eq. (6) (i.e., a point estimation) has the same form as that of existing frameworks in Eq. (3) or (4), our framework can benefit from the decoupled nature of the optimization, that is, θ_{D} and θ_{T} are optimized individually in Eq. (5) and Eq. (6), respectively. In Eq. (5), $\mathcal{R}$ may be used solely for regularizing the behavior of f_{D}. In contrast, in Eq. (6), $\mathcal{R}$ is minimized for selecting θ_{T}. Such a clear distinction of the semantics of $\mathcal{R}$ in each scene can make the tuning of λ easier. Moreover, the decoupled optimization allows us to use different optimizers and datasets for estimating θ_{D} and θ_{T}.

If we should estimate different θ_{T} for each query x, we have two options: One is to solve Eq. (6) with S'_X being the singleton, $S'_X = \{x\}$, which is inefficient when there are a large number of queries. Another option is to solve

$$\theta_h^* \in \arg\min_{\theta_h} \sum_x \mathcal{R}_{S'_X}(\theta_{\mathrm{T}} = h(x), \theta_{\mathrm{D}}^*), \tag{7}$$

where $h\colon \mathcal{X} \to \Theta_{\mathrm{T}}$ is a trainable model parameterized with θ_h and is used for inferring θ_{T} given x in an amortized manner (i.e., so-called an encoder).

Distribution Estimation of θ_{T} (optional) Since only $\mathcal{R}$ dictates the choice of θ_{T}, it is reasonable to define θ_{T}'s posterior distribution as

$$p(\theta_{\mathrm{T}} \mid \text{data}) \propto p(\theta_{\mathrm{T}}) \exp\{-\beta \mathcal{R}_{S'_X}(\theta_{\mathrm{T}}, \theta_{\mathrm{D}}^*)\}, \tag{8}$$

for some $\beta > 0$. Even though the normalizing constant of $p(\theta_{\mathrm{T}} \mid \text{data})$ cannot be computed, we can draw samples from the distribution using MCMC.

3.3 Technical Notes

In **Step 1**, we extended f_{D} so that it works adaptively to different θ_{T}s. This increased model complexity does not come with an additional need for real training data because we can draw as many random samples of θ_{T} as the computational resources allow in the estimation process. As a result, while the learning problem becomes (hopefully slightly) more complicated than the original ones, it would not make the problem significantly more challenging. We will empirically confirm it through numerical experiments.

Though the full applicability of the proposed framework is limited to moderate-dimensional θ_{T}, in practice, it would not significantly limit applications because theory-based models often have (or should have) a small number of parameters. Models with high-dimensional θ_{T} such as unknown fields are a challenging open problem.

As repeatedly argued, the main characteristic of the proposed formulation is that it makes it easy to analyze $\mathcal{R}$s empirically. Here, care should be taken not to reuse the same data both for such an analysis and the estimation with $\mathcal{R}$ selected via the former analysis. We actually commit "data reuse" in parts of the experiments in Section 5, which is admittable only because our purpose is not to analyze the data but to compare the different estimation schemes.

3.4 A Numerical Example

We illustrate how the framework could be used with a trivial numerical example. We generate data from

$$y = \sin(x) + \cos(x) + e,$$

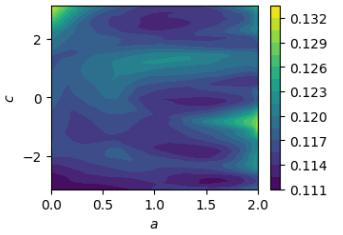

Figure 2: Test prediction RMSE of the numerical example for f_{D} trained by Eq. (5) with $\lambda = 0$ on the toy dataset in Section 3.4. The horizontal and vertical axes correspond to a and c of f_{T}, respectively.

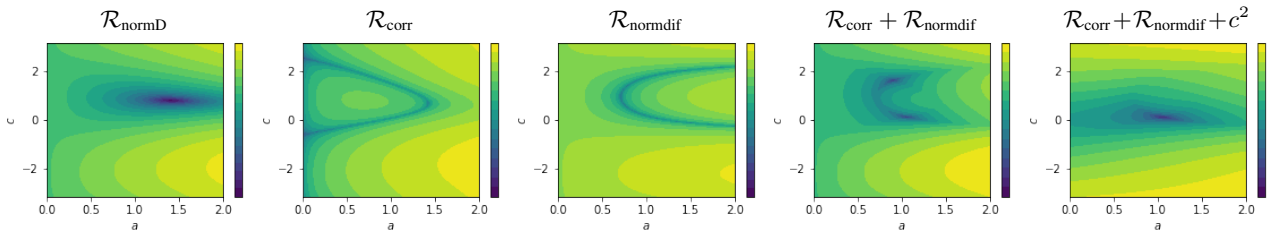

Figure 3: Landscapes of some regularizers and their combinations for f_{D} trained by Eq. (5) with $\lambda = 0$ on the toy dataset in Section 3.4. The horizontal and vertical axes correspond to a and c of f_{T}, respectively.

where $x \in \mathcal{X} = [-\pi, \pi]$, and $e \sim \mathcal{N}(0, 0.1^2)$. The model we use is a deep grey-box model:

$$\mathcal{C}(f_{\text{T}}, f_{\text{D}}; x) = f_{\text{T}}(x) + f_{\text{D}}(f_{\text{T}}, x),$$
$$f_{\text{T}}(x) = a \sin(x + c),$$
$$f_{\text{D}}(f_{\text{T}}, x) = \text{MLP}(x, \theta_{\text{T}}, f_{\text{T}}(x)),$$

where $\theta_{\text{T}} = [a, c]$ is the f_{T}'s parameters. $\text{MLP}(\cdot)$ is a feed-forward neural net. We feed the MLP with $[x; \theta_{\text{T}}; f_{\text{T}}(x)] \in \mathbb{R}^4$, and it returns values in $\mathbb{R}$. Note that our f_{D} is aware of θ_{T} and can work adaptively to different values of θ_{T}, as advised in **Step 1** (Section 3.2.1).

The next step, **Step 2** (Section 3.2.2), is to optimize the expected objective in Eq. (5). We run it with the squared error

$$\mathcal{L} = \sum (y - \mathcal{C}(f_{\text{T}}, f_{\text{D}}; x))^2$$

and $\lambda = 0$ (i.e., we do not specify $\mathcal{R}$). We let $p(\theta_{\text{T}})$ be the uniform distribution over $[0, 2] \times [-\pi, \pi]$. Figure 2 shows the value of the prediction error, $\mathcal{L}$, on test data for different values of $\theta_{\text{T}} = [a, c]$. Although some extreme values of θ_{T} tend to have slightly larger errors, we can see that the model predicts similarly well with different θ_{T}s.

As **Step 3** (Section 3.2.3), we overview the landscapes of some candidates of $\mathcal{R}$. In Fig. 3, we visualize the values of the following $\mathcal{R}$s over a grid of $\theta_{\text{T}} = [a, c]$:

- $\mathcal{R}_{\text{normD}} = \sum_x f_{\text{D}}(x)^2$ (i.e., f_{D} should work minimally);
- $\mathcal{R}_{\text{corr}} = |\sum_x f_{\text{T}}(x) \cdot f_{\text{D}}(x)|$ (i.e., f_{T} and f_{D} should work uncorrelatedly); and
- $\mathcal{R}_{\text{normdif}} = \left|\sum_x f_{\text{T}}(x)^2 - \sum_x f_{\text{D}}(x)^2\right|$ (i.e., f_{T} and f_{D} should work to the same extent).

The leftmost heatmap shows $\mathcal{R}_{\text{normD}}$; we see that it reaches the minimum around $[a, c] = [\sqrt{2}, \pi/4]$, which is natural because $\sin(x) + \cos(x) = \sqrt{2}\sin(x + \pi/4)$. It serves as a "truth" for those who believe that f_{D} should work minimally. However, the case is not over if one has prior knowledge dictating that f_{T} and f_{D} should be uncorrelated. Such a user would like to use $\mathcal{R}_{\text{corr}}$, which is visualized in the second heatmap from the left. We see that multiple θ_Ts are practically equivalent under this regularizer. If we want to further narrow down the possible choices of θ_{T}, we need to consider more criteria. Let us see $\mathcal{R}_{\text{normdif}}$, which is shown in the center heatmap. We find that it is not enough, either, to select a (few) θ_{T} value(s). Summing $\mathcal{R}_{\text{corr}}$ and $\mathcal{R}_{\text{normdif}}$ results in the landscape shown in the fourth heatmap from the left, where we see two minima. In practice, this would be the most reasonable regularizer that could be designed and explained through these empirical analyses of $\mathcal{R}$s.

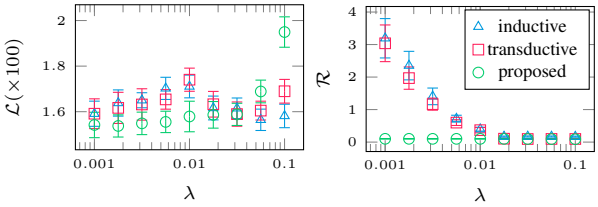

Figure 4: Main loss $\mathcal{L}$ and regularizer $\mathcal{R}$ on test data for the models learned with different values of regularization hyperparameter, λ, on the toy dataset in Section 3.4. Error bars show the standard errors by 20 random trials.

We provide one more visualization in the rightmost panel of Fig. 3, just for completeness. It shows the values of $\mathcal{R}_{\text{corr}} + \mathcal{R}_{\text{normdif}} + c^2$, which has a single extremum. However, the c^2 term can hardly be specified in practice without the knowledge of the data-generating process.

A message of the above story is that we should analyze the landscape of $\mathcal{R}$, instead of uncritically optimizing it. Suppose that we follow the prior knowledge dictating that f_{T} and f_{D} should be uncorrelated. Applying a standard learning scheme (i.e., optimize $\mathcal{L} + \lambda\mathcal{R}_{\text{corr}}$ without the adaptivity of f_{D}) would result in θ_{T} being practically a random choice from one of the many local minima of $\mathcal{R}_{\text{corr}}$ that we see in Fig. 3 (2nd from the left). This is hardly meaningful since these parameters are supposed to have an interpretation within the context of the relevant domain knowledge.

Let us take one of the further options of **Step 3**, the point estimation. Suppose using $\mathcal{R} = \mathcal{R}_{\text{corr}} + \mathcal{R}_{\text{normdif}} + c^2$. We compare our framework, Eqs. (5) and (6), with inductive learning, Eq. (3), and transductive learning, Eq. (4). We vary the value of λ from 0.001 to 0.1. Figure 4 reports the squared error, $\mathcal{L}$, and the regularizer, $\mathcal{R}$, computed on a test set for each configuration. Our framework achieves small values of $\mathcal{R}$ practically for any value of λ; this is not the

case for inductive and transductive learning. The slightly large $\mathcal{L}$ for the proposed framework with $\lambda = 0.1$ is probably because the training of the adaptive f_{D} was inhibited by the large value of λ. This result supports the use of a small value of λ in the optimization of Eq. (5). We also confirmed that our framework worked well with $\lambda = 0$.

4 RELATED WORK

Deep Grey-box Modeling Combination of deep neural nets and theory-driven models has been studied in various contexts (e.g., recent studies include Azari et al., 2020; Bikmukhametov and Jäschke, 2020; E et al., 2020; Karniadakis et al., 2021; Kon et al., 2022; Mehta et al., 2021; Reichstein et al., 2019; Takeishi and Kalousis, 2021; von Rueden et al., 2021; Wang, 2021; Wehenkel et al., 2023; Willard et al., 2020; Yin et al., 2021); we cannot enumerate more from a large number of studies in older times and in application domains, so readers are recommended consulting the references in the aforementioned papers.

Learning Schemes Our method can minimize a part of the objective at prediction time. This is also the case with model-agnostic meta-learning (MAML) (Finn et al., 2017). Differently from our setting, MAML is meant for settings where one has many tasks and wants to adapt to a new task at prediction time. Meta-tailoring (Alet et al., 2021) can be thought of as a variant of MAML with each sample being one task. Nonetheless, these methods do not necessarily allow us to efficiently explore objectives' landscape.

Seo et al. (2021) proposed a method to learn deep neural nets with supervision from both data and rules. They suggest "marginalizing out" a parameter during training so that we can choose it freely in the inference time. Despite such similarities, there are several differences from our framework, e.g., in the target of the marginalization and in how theory is incorporated, which prevent direct comparison.

5 EXPERIMENTS

We demonstrate[2] the capability of the proposed framework to explore the landscape of regularizers. We also compare the proposed framework with existing learning schemes supposing we take the option of point estimation.

5.1 Datasets, Models, and Optimization

In all cases, $\mathcal{L}$ is the mean squared error. See the appendix for more details of the experimental settings.

Controlled Pendulum We used time-series data of a frictionless compound pendulum controlled by an unknown regulator. The dataset comprises pairs

$$(x = s_t, y = (s_{t+1}, \ldots, s_{t+10})),$$

where $s_t = [s_t^{(1)}, s_t^{(2)}]$ is the state (angle $s_t^{(1)}$ and angular velocity $s_t^{(2)}$) of the pendulum at time t. We built the dataset by gathering such pairs from different sequences with different initial conditions.

The model we used is

$$\mathcal{C}(f_{\mathrm{T}}, f_{\mathrm{D}}; x) = \mathrm{ODESolve}\left[\dot{s} = f_{\mathrm{T}}(s) + f_{\mathrm{D}}(s, \theta_{\mathrm{T}}, f_{\mathrm{T}}(s)) \;\middle|\; s_0 = x \right],$$

where f_{T} is the equation of motion of a compound pendulum without a controller:

$$f_{\mathrm{T}}(s) = \left[s^{(2)}, \frac{3\theta_{\mathrm{T}}}{2} \sin(s^{(1)}) \right],$$

and $f_{\mathrm{D}}(s, \theta_{\mathrm{T}}, f_{\mathrm{T}}(s))$ is a neural net with fully-connected layers. So, f_{D} should mimic the behavior of the unknown controller. It is not obvious which $\mathcal{R}$ we should use because we do not know the controller's nature. In Eq. (5), we set $\mathcal{R} = \mathcal{R}_{\mathrm{normD}} + \mathcal{R}_{\mathrm{corr}}$, where $\mathcal{R}_{\mathrm{normD}} = \sum_x \|f_{\mathrm{D}}(x)\|_2^2$ and $\mathcal{R}_{\mathrm{corr}} = |\sum_x f_{\mathrm{T}}(x) \cdot f_{\mathrm{D}}(x)|$ are candidate $\mathcal{R}$s, with a small coefficient $\lambda = 0.001$.

Reaction–Diffusion System We generated data from the two-dimensional two-component reaction–diffusion system of the FitzHugh–Nagumo type: $\partial u/\partial t = 0.0015\Delta u + u - u^3 - v + 0.005$ and $\partial v/\partial t = 0.005\Delta v + u - v$, where Δ is the Laplace operator. We discretized the space such that $u, v \in \mathbb{R}^{32\times 32}$. The dataset comprises pairs $(x = s_0, y = (s_1, s_2, \ldots, s_{15}))$, where $s_t = [u_t, v_t]$ is the state at time t. We note that similar configurations of the same system have been used in previous related studies (Wehenkel et al., 2023; Yin et al., 2021).

We used a grey-box neural ODE as in the previous example. The theory-driven part is from the diffusion term of the true data-generating process, that is,

$$f_{\mathrm{T}}(s) = [a\Delta u, b\Delta v],$$

where $\theta_{\mathrm{T}} = [a, b]$ is the parameter. The data-driven part is $f_{\mathrm{D}}(s, \theta_{\mathrm{T}}) = \mathrm{ConvNet}_{\theta_{\mathrm{T}}}(s)$, a convolutional net whose filter is partially parameterized by θ_{T}. In the previous studies

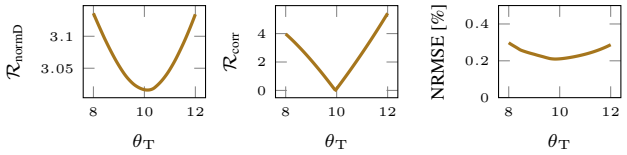

Figure 5: (*Left* & *center*) Landscapes of the regularization terms, $\mathcal{R}_{\mathrm{normD}}$ and $\mathcal{R}_{\mathrm{corr}}$ computed on the test set of the **controlled pendulum** dataset. (*Right*) Test NRMSEs.

[2]The codes are available at `https://github.com/n-takeishi/deepgreybox`.

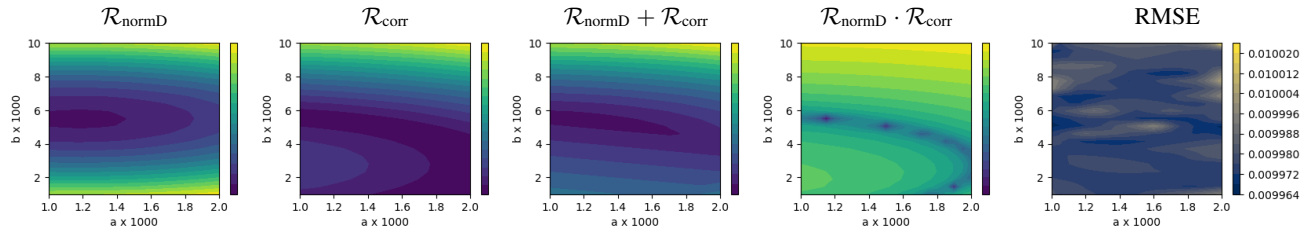

Figure 6: (*Left four*) Landscapes of $\mathcal{R}_{\text{normD}}$, $\mathcal{R}_{\text{corr}}$, and their combinations, for the **reaction–diffusion system** dataset. The horizontal and vertical axes correspond to a and b of f_{T}, respectively. (*Right*) Test RMSEs.

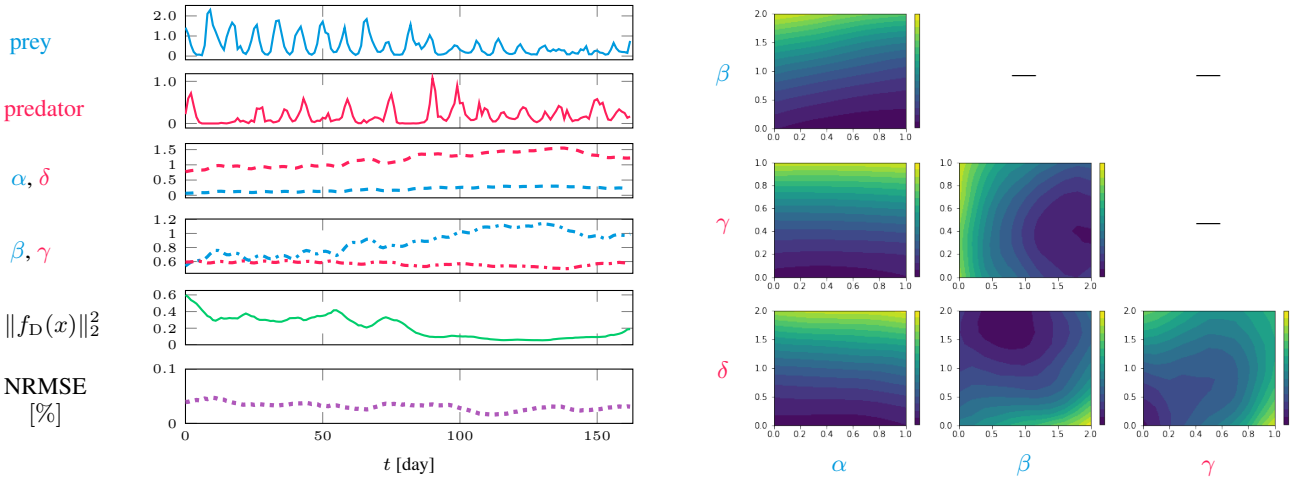

Figure 7: (*Left*) A test sequence of the **predator–prey system** dataset, inferred $\alpha, \beta, \gamma, \delta$, $\|f_{\text{D}}(x)\|_2^2$, and prediction NRMSE. (*Right*) Partial landscape of $\mathcal{R} = \sum_x \|f_{\text{D}}(x)\|_2^2$ at $t = 65$.

(Wehenkel et al., 2023; Yin et al., 2021), it has been reported that $\mathcal{R}_{\text{normD}} = \sum_x \|f_{\text{D}}(x)\|_2^2$ works to some extent, yet without detailed analysis of its behavior. Hence, we use $\mathcal{R} = \mathcal{R}_{\text{normD}} + \mathcal{R}_{\text{corr}}$ with a small value of λ to examine both regularizers as candidates.

Predator–Prey System We used real data of a planktonic predator–prey system (Blasius et al., 2020). From the original data, we extracted the measurements of the population densities of the prey (unicellular algae) and the predator (rotifer). We split the original time-series into subsequences of length 11 [days] and created a dataset comprising $x = y = (s_t, \ldots, s_{t+10})$ where $s_t = [s_t^{(1)}, s_t^{(2)}] \in \mathbb{R}^2$ is the state (prey density $s_t^{(1)}$ and predator density $s_t^{(2)}$) at day t. Hence, the task is autoencoding.

The model comprises a decoder and two encoders. The decoder is again a grey-box neural ODE:

$$\mathcal{C}(f_{\text{T}}, f_{\text{D}}; x) = \text{ODESolve}\left[\dot{s} = f_{\text{T}}(s) + f_{\text{D}}(z, s, \theta_T, f_{\text{T}}(s)) \;\middle|\; s_0 = x_0 \right],$$

where z is a latent variable of the data-driven part. The theory-driven part is taken from Lotka–Volterra equations, that is,

$$f_{\text{T}}(s) = [\alpha s^{(1)} - \beta s^{(1)} s^{(2)},\ -\gamma s^{(2)} + \delta s^{(1)} s^{(2)}],$$

where $\theta_{\text{T}} = [\alpha, \beta, \gamma, \delta]$ (α: prey's growth rate, β: prey's decay rate, γ: predator's decay rate, and δ: predator's growth rate) is the set of the parameters. f_{D} is a neural net with fully-connected layers.

In addition to the decoder $\mathcal{C}$, the model has two encoders. One of them is a neural net that takes x as input and outputs z. Another is a neural net that takes x and infers θ_{T}. We run Eq. (5) with $\mathcal{R} = \sum_x \|f_{\text{D}}(x)\|_2^2$ and $\lambda = 0.001$ to estimate θ_{D} and the first encoder's parameters, by which we can explore the $\mathcal{R}$'s landscape. Afterward, we run Eq. (7) to learn the second encoder.

5.2 Exploration of the Landscape of $\mathcal{R}$

Figure 5 shows, for the **controlled pendulum** dataset, the values of $\mathcal{R}_{\text{normD}}$ and $\mathcal{R}_{\text{corr}}$ for different θ_{T}s. We can see that the two regularizers have slightly different peaks. While it is not easy to decide which one is better in some sense, it is notable that our method can provide such empirical clues on the differences between the regularizers. The rightmost plot of Fig. 5 shows the normalized root mean

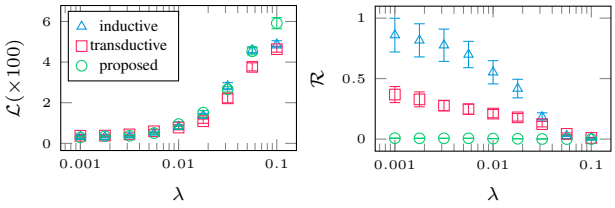

Figure 8: Loss $\mathcal{L}$ and regularizer $\mathcal{R}$ on test set of the **controlled pendulum** dataset. Error bars show the standard errors by 20 random trials.

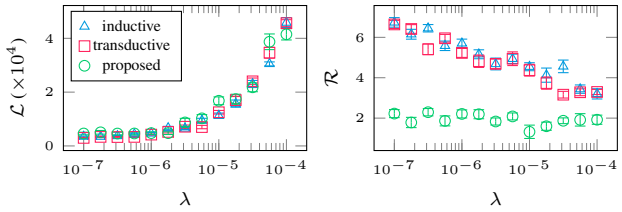

Figure 9: Loss $\mathcal{L}$ and regularizer $\mathcal{R}$ on test set of the **reaction–diffusion** system dataset. Error bars show the standard errors by 20 random trials.

squared errors (NRMSEs) of the prediction on the test set. The model can predict well regardless of the value of θ_{T}.

Figure 6 visualizes the $\mathcal{R}$ candidates and the test RMSE for the **reaction–diffusion system** dataset, analogously to what was explained in the previous paragraph. $\mathcal{R}_{\mathrm{normD}}$ has a peak at some value, while the extrema of $\mathcal{R}_{\mathrm{corr}}$ are much less clear. We found that the product of these two terms, $\mathcal{R}_{\mathrm{normD}} \cdot \mathcal{R}_{\mathrm{corr}}$, also have clear, if not unique, peaks. Interestingly, one of the local minima of $\mathcal{R}_{\mathrm{normD}} \cdot \mathcal{R}_{\mathrm{corr}}$ nicely points the data-generating value of $\theta_{\mathrm{T}} = [a, b]$, that is, $[a, b] = [0.0015, 0.005]$. On the other hand, $\mathcal{R}_{\mathrm{normD}}$, which has been used in previous studies, does not point to this value as the minimum. This suggests that, again, we should empirically analyze $\mathcal{R}$ to know how different $\mathcal{R}$ may result in different θ_{T} estimations, instead of optimizing it uncritically. We are *not* suggesting $\mathcal{R}_{\mathrm{normD}} \cdot \mathcal{R}_{\mathrm{corr}}$ is a "better" regularizer; we never know the truth of θ_{T} in practice and thus cannot evaluate anything from such a viewpoint.

Figure 7 (left) shows an outcome of the estimated model on a test sequence of the **predator–prey system** dataset. We took a sliding window of length 11 from the long test sequence, applied the model to the subsequences from the window, and calculated the averages of the outputs for the overlapping steps. The value of $\|f_{\mathrm{D}}(x)\|_2^2$ at each time is informative to assess how well the f_{T} could explain the data. We can observe that the latter half of the sequence is relatively well explained by f_{T}. We can also analyze the landscape of the regularizer, $\mathcal{R} = \sum_x \|f_{\mathrm{D}}(x)\|_2^2$. Figure 7 (right) shows the slices of the values of the $\mathcal{R}$. Each slice is computed by fixing the values of the remaining elements of θ_{T} at the encoder's outputs. An observation is that while $\mathcal{R}$ seems to have some extrema along the directions of β, γ, δ, it looks quite flat along the direction of α. It implies the possible difficulty of inferring α, prey's growth rate without interaction, from the data, and thus we should not immediately interpret at least the results of α's inference.

5.3 Point Estimation of θ_{T}

We performed one of the further options in the proposed framework. For the **controlled pendulum** dataset, we set the regularizer as $\mathcal{R} = \mathcal{R}_{\mathrm{corr}}$ and performed the point estimation by Eq. (3) (inductive), Eq. (4) (transductive), and Eq. (6) (proposed). Figure 8 shows the values of $\mathcal{L}$ and $\mathcal{R}$ on the test set. $\mathcal{L}$ is not well minimized with a large λ for all the methods, which is natural. A notable difference is that by the proposed method, $\mathcal{R}$ is well minimized even with small λ, though it is not surprising because the optimization is decoupled. In contrast, the inductive and transductive learning schemes result in larger $\mathcal{R}$ values with small λs, due to which striking a good value of λ may be difficult for these learning schemes. Figure 9 reports basically the same thing but for the **reaction–diffusion system** dataset.

6 CONCLUSION

Deep grey-box models are combinations of deep neural networks and theory-driven models. We argued that, toward trustworthy estimation of the theory-driven model, we should empirically analyze the regularizers. We proposed a modification of the network and the optimization objective such that we can empirically analyze regularizers without re-training the model. The proposed formulation is also useful as it decouples the training of the data-driven and theory-driven models. The main limitation is the applicability to theory-driven models with high-dimensional parameters. They pose issues in two aspects: computation of the expectation with regard to the theory parameters and visualization of a regularizer's landscape.

Acknowledgements

This work was supported by the Innosuisse project *Industrial artificial intelligence for intelligent machines and manufacturing digitalization* (39453.1 IP-ICT) and the Swiss National Science Foundation Sinergia project *Modeling pathological gait resulting from motor impairments* (CRSII5_177179).

References

Alet, F., Bauza, M., Kawaguchi, K., Kuru, N. G., Lozano-Perez, T., and Kaelbling, L. P. (2021). Tailoring: Encoding Inductive Biases by Optimizing Unsupervised Objectives at Prediction Time. In *Advances in Neural Information Processing Systems 34*, 29206–29217.

Arık, S. Ö., Li, C.-L., Yoon, J., Sinha, R., Epshteyn, A., Le, L. T., Menon, V., Singh, S., Zhang, L., Yoder, N., Nikoltchev, M., Sonthalia, Y., Nakhost, H., Kanal, E., and Pfister, T. (2020). Interpretable Sequence Learning for COVID-19 Forecasting. In *Advances in Neural Information Processing Systems 33*, 18807–18818.

Azari, A. R., Lockhart, J. W., Liemohn, M. W., and Jia, X. (2020). Incorporating Physical Knowledge into Machine Learning for Planetary Space Physics. In *Frontiers in Astronomy and Space Sciences* 7, 36.

Bikmukhametov, T. and Jäschke, J. (2020). Combining Machine Learning and Process Engineering Physics towards Enhanced Accuracy and Explainability of Data-Driven Models. In *Computers & Chemical Engineering* 138, 106834.

Blasius, B., Rudolf, L., Weithoff, G., Gaedke, U., and Fussmann, G. F. (2020). Long-Term Cyclic Persistence in an Experimental Predator–Prey System. In *Nature* 577, 226–230.

Breiman, L. (2001). Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author). In *Statistical Science* 16.3, 199–231.

E, W., Han, J., and Zhang, L. (2020). Integrating Machine Learning with Physics-Based Modeling. arXiv: 2006.02619.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 1126–1135.

Gammerman, A., Vovk, V., and Vapnik, V. (1998). Learning by Transduction. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 148–155.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-Informed Machine Learning. In *Nature Reviews Physics* 3, 422–440.

Kon, J., Bruijnen, D., van de Wijdeven, J., Heertjes, M., and Oomen, T. (2022). Unifying Model-Based and Neural Network Feedforward: Physics-guided Neural Networks with Linear Autoregressive Dynamics. arXiv: 2209.12489.

Mehta, V., Char, I., Neiswanger, W., Chung, Y., Nelson, A. O., Boyer, M. D., Kolemen, E., and Schneider, J. (2021). Neural Dynamical Systems: Balancing Structure and Flexibility in Physical Prediction. In *Proceedings of the 2021 IEEE Conference on Decision and Control*, 3735–3742.

Qian, Z., Zame, W. R., Fleuren, L. M., Elbers, P., and van der Schaar, M. (2021). Integrating Expert ODEs into Neural ODEs: Pharmacology and Disease Progression. In *Advances in Neural Information Processing Systems 34*, 11364–11383.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. In *Journal of Computational Physics* 378, 686–707.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., and Prabhat (2019). Deep Learning and Process Understanding for Data-Driven Earth System Science. In *Nature* 566.7743, 195–204.

Sasaki, R., Takeishi, N., Yairi, T., and Hori, K. (2019). Neural Gray-Box Identification of Nonlinear Partial Differential Equations. In Lecture Notes in Computer Science 11671, 309–321.

Schnell, P., Holl, P., and Thuerey, N. (2022). Half-Inverse Gradients for Physical Deep Learning. In *Proceedings of the 10th International Conference on Learning Representations*.

Seo, S., Arik, S. O., Yoon, J., Zhang, X., Sohn, K., and Pfister, T. (2021). Controlling Neural Networks with Rule Representations. In *Advances in Neural Information Processing Systems 34*, 11196–11207.

Sohlberg, B. and Jacobsen, E. (2008). Grey Box Modelling – Branches and Experiences. In *IFAC Proceedings Volumes* 41.2, 11415–11420.

Takeishi, N. and Kalousis, A. (2021). Physics-Integrated Variational Autoencoders for Robust and Interpretable Generative Modeling. In *Advances in Neural Information Processing Systems 34*, 14809–14821.

Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., Walczak, M., Garcke, J., Bauckhage, C., and Schuecker, J. (2021). Informed Machine Learning – A Taxonomy and Survey of Integrating Knowledge into Learning Systems. In *IEEE Transactions on Knowledge and Data Engineering*.

Wang, R. (2021). Physics-Guided Deep Learning for Dynamical Systems: A Survey. arXiv: 2107.01272.

Wehenkel, A., Behrmann, J., Hsu, H., Sapiro, G., Louppe, G., and Jacobsen, J.-H. (2023). Robust Hybrid Learning with Expert Augmentation. In *Transactions on Machine Learning Research*, 470.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V. (2020). Integrating Physics-Based Modeling with Machine Learning: A Survey. arXiv: 2003.04919.

Yin, Y., Le Guen, V., Dona, J., de Bézenac, E., Ayed, I., Thome, N., and Gallinari, P. (2021). Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting. In *Proceedings of the 9th International Conference on Learning Representations*.

A EXPERIMENTAL SETTINGS

A.1 Common

Optimization In all the experiments including the one in Section 3.4, the objective of Eq. (5) and its gradients were estimated using only one sample of θ_{T} per an instance of x, which we denote by $m = 1$. We also tried larger m up to $m = 100$ per a single x, but the final improvement of performance was marginal. We used the `AdamW` optimizer for the optimization of inductive learning, Eq. (3), transductive learning, Eq. (4), and the training-time optimization of the proposed method, Eq. (5). The parameters of the optimizer were set to the default values of the library unless stated otherwise.

Others

- All implementations were done with `PyTorch 1.11.0`.
- The NRMSEs were computed by dividing RMSEs by $y_{\max} - y_{\min}$.

A.2 Toy Dataset (in Section 3.4)

Data In the example in Section 3.4, we generated the data by sampling x from the uniform distribution on $\mathcal{X} = [-\pi, \pi]$. We created training, validation, and test datasets, each of which was with 40 samples of (x, y).

Model In the model, f_{D} is a neural net with fully-connected layers (with two hidden layers of size 16) and the leaky ReLU activation function.

Optimization We optimized the model parameters ($\theta_{\mathrm{T}} = [a, c]$ and θ_{D} in the inductive and transductive learning schemes, Eqs. (3) and (4); only θ_{D} in the proposed method, Eq. (5)) with learning rate varied from 0.01 to 0.0001 exponentially. We set the mini-batch size to 10 and ran the optimization for 2000 epochs. In the inductive and transductive learning schemes, the value of $\theta_{\mathrm{T}} = [a, c]$ was constrained to be $0 \leq a \leq 2$ and $-\pi \leq c \leq \pi$. In the training-time optimization of the proposed method, Eq. (5), we sampled $\theta_{\mathrm{T}} = [a, c]$ from the uniform distribution on $0 \leq a \leq 2$ and $-\pi \leq c \leq \pi$. The prediction-time optimization of the proposed method, Eq. (6), was performed with the Adam optimizer (without weight decay) with learning rate varied from 0.01 to 0.0001. We ran it for 2000 epochs with the full batch.

A.3 Controlled Pendulum

Data We used the dataset available[3] on the GitHub repository of the `Stable Baselines` package[4]. It is a collection of sequences measuring the state of a frictionless compound pendulum, with a rigid uniform rod of length 1. The gravitational acceleration is $g = 10$ in the data-generation environment. In each sequence, the pendulum starts at a random position and is controlled to stay vertically upright by some controller. The detail of the controller does not matter, so we just leave it unknown. The original dataset comprises sequences of length 200. We disposed of the last 100 steps of each sequence because most of the time, the pendulum is already almost stabilized at the goal state by that time. We ran a sliding window of size 11 on the remaining sequences and created pairs (x, y) with x being the first snapshot in the window and y being the last 10 snapshots in the window. We split the original dataset into training, validation, and test sets of size 3600, 2700, and 2700, respectively.

Model In the model, the neural net, $\mathrm{MLP}(t, s, \theta_{\mathrm{T}}, f_{\mathrm{T}}(s))$, is a feed-forward network with fully-connected layers with three hidden layers of size 128. It takes the concatenation of s, θ_{T}, and $f_{\mathrm{T}}(s)$ as the input. We used the `torchdiffeq` library[5] for the numerical integration of the ODE with the 4th-order Runge–Kutta method.

Optimization In training, we varied the learning rate from 0.001 to 0.00001 exponentially, set the mini-batch size to 50, and ran the optimization for 500 epochs. In the inductive and transductive learning schemes, the value of θ_{T} was constrained to be in $[8, 12]$. In the training-time optimization of the proposed method, Eq. (5), we sampled θ_{T} from the uniform distribution on $[8, 12]$. The prediction-time optimization of the proposed method, Eq. (6), was performed simply by the grid search because θ_{T} is one-dimensional.

[3] direct link to the GitHub repository

[4] `https://github.com/Stable-Baselines-Team/stable-baselines/`

[5] `https://github.com/rtqichen/torchdiffeq/`

A.4 Reaction–Diffusion System

Data We generated the data from the two-dimensional two-component reaction–diffusion system of the FitzHugh–Nagumo type:

$$\frac{\partial u}{\partial t} = 0.0015\Delta u + u - u^3 - v + 0.005,$$
$$\frac{\partial v}{\partial t} = 0.005\Delta v + u - v.$$

The two variables of the system, u and v, are defined over the two-dimensional space spanning $[-1, 1] \times [-1, 1]$. We discretized the space with the 32×32 even grid. In generating the data and in the trained model, the Laplacian operator, Δ, was computed using the five-point stencil. We generated 1000 sequences of length 16 from $t = 0$ to $t = 1.5$ using the 4th-order Runge–Kutta method with step size 0.001. We prepared training, validation, and test sets of size 400, 300, and 300, respectively.

Model In the model, the neural net, $\mathrm{ConvNet}_{\theta_\mathrm{T}}(s)$, returns the sum of the outputs of two subnetworks. One of them is a convolutional network with two hidden layers having 16 channels and with the leaky ReLU activation function. Another subnetwork is a two-layer convolutional network whose filters' weights are the discrete Laplacian operator (with the five-point stencil) multiplied by scalars for each filter. Those scalars are the output of a feed-forward network that takes θ_T as input. It has two hidden layers of size 128 with the leaky ReLU activation function applied to the intermediate layers and the hyperbolic tangent applied to the final layer. When the output of this network is below $-\theta_\mathrm{T}/2$, it is clipped to be $-\theta_\mathrm{T}/2$ so that the second subnetwork does not completely cancel the output of f_T.

Optimization In training, we fixed the learning rate at 0.001, set the mini-batch size to 20, and ran the optimization for 1000 epochs. In the inductive and transductive learning schemes, the value of $\theta_\mathrm{T} = [a, b]$ was constrained to be in $[0.001, 0.002] \times [0.001, 0.01]$. In the training-time optimization of the proposed method, Eq. (5), we sampled θ_T from the uniform distribution on $[0.001, 0.002] \times [0.001, 0.01]$. The prediction-time optimization of the proposed method, Eq. (6), was performed with the Adam optimizer (without weight decay) with learning rate 0.001. We ran it for 100 epochs with the full batch.

A.5 Predator–Prey System

Data We used the data available online[6], which contain measurements of the population densities of the prey (unicellular algae) and the predator, (rotifer). In order to adjust the scale of the data values, we roughly normalized the data by multiplying 0.5 and 0.02 by the population densities of the two species, respectively. We excluded the part of data containing a value larger than 7 (after the normalization) from the training and validation sets; the test set might include such parts, but we did not check it.

Model The model has three neural networks in total: f_D, the encoder of z, and the encoder of θ_T. These networks have the same architecture: three hidden layers of size 128 and the leaky ReLU activation function.

Optimization In training, we fixed the learning rate at 0.001, set the mini-batch size to 100, and ran the optimization for 500 epochs. In the inductive and transductive learning schemes, the value of $\theta_\mathrm{T} = [\alpha, \beta, \gamma, \delta]$ was constrained to be in $[0, 1.5] \times [0, 3] \times [0, 1.5] \times [0, 3]$. In the training-time optimization of the proposed method, Eq. (5), we sampled θ_T from the uniform distribution on $[0, 1.5] \times [0, 3] \times [0, 1.5] \times [0, 3]$. The prediction-time optimization of the proposed method, Eq. (6), was performed with the Adam optimizer (without weight decay) with learning rate 0.001. We ran it for 50 epochs with the mini-batch of size 100.

[6] https://doi.org/10.6084/m9.figshare.10045976.v1