O_2PDE: AN END-TO-END FRAMEWORK FOR SOLVING INVERSE PDE PROBLEMS FROM OBSERVATIONS ALONE

Anonymous authors
Paper under double-blind review

ABSTRACT

Solving inverse PDE problems from observations remains a fundamental challenge in applied mathematics and physics-informed machine learning. Classical approaches, such as system identification and symbolic regression, provide interpretable and closed-form representations but are typically restricted to expert-level skills. Recent advances in machine learning have exhibited promise, but the performance of existing solvers relies heavily on abundant labeled data and domain expert priors, thus hindering practical applications in mathematics and physics due to a lack of accuracy and robustness. This work considers the challenging task of reconstructing the PDE system from observations alone without any domain expert priors. We propose an end-to-end framework, named O_2PDE, for tackling this challenge. The workflow of O_2PDE consists of four modules: Data Encoder that converts observations into latent representations, Function Encoder that encodes symbolic candidate functions to queries, Decoder with Cross-Attention for conditioning queries on observation representations, and Dual Head that jointly predicts structure and coefficient identifications. The whole model is supervised by a combination loss relative to multi-label classification and coefficient regression, with a regularizer that fuses temporal chunks and consistency losses for enforcing prediction consistency across chunks. We conduct experiments on a collection of PDE systems to evaluate the effectiveness and robustness of O_2PDE in comparison with six baselines, which contains 1.65 million PDE systems and 458.11 GB observable data. Compared to six baselines, our method achieves the best performance, where O_2PDE improves 91.9% F1 and 52.42% EM and reduces 94.8% L2RE rather than the fine-tuned LLMs, with only 5.1% of the LLM's total parameters. Even corrupting 5% and 10% Gaussian noises to observations, O_2PDE improves F1 by 21.7% and 18.1%, while reducing L2RE by 56.4% and 47.1%, respectively, compared to the SOTA baseline. This is a laudable result, not only confirming the reconstruction advances of O_2PDE but also opening the door of developing foundation models for inverse PDE problem solving.

1 INTRODUCTION

The inverse Partial Differential Equations (PDEs) problem is a long-standing issue confronted in applied mathematics and physics-informed machine learning (Berg & Nyström, 2019; Karniadakis et al., 2021), where the evolution of a physical system is governed by underlying PDEs that usually remain unknown and only observations are available. Such problems arise broadly in scientific discovery and engineering, such as earth-system modeling (Zanna & Bolton, 2020; Huntingford et al., 2025) where the reconstructed equations are used for revealing underlying climate mechanisms, and cancer biology (Srivastava et al., 2025) that explicitly governs cancer-cell migration and proliferation. A fundamental and significant challenge of solving inverse PDE problems is how to accurately and robustly reconstruct the complete PDE system, including both its structure and coefficients, based solely on observations without relying on any explicit prior knowledge from domain experts.

Past decades have witnessed a lot of efforts on tackling this challenge. Traditional techniques contain the classical system identification (Lataire et al., 2012; de Rozario & Oomen, 2018; Kuntsevich, 1996; Li et al., 2010; Verhaegen & Yu, 1995; Hallemans et al., 2021; Chen et al., 2020), grounded

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

in rigorous mathematical theory and full observability. Symbolic regression and its variants (Rudy et al., 2017; Zhang & Lin, 2018; He et al., 2022; Xu et al., 2021; Wang et al., 2019; Gurevich et al., 2019; Messenger & Bortz, 2021) can recover closed-form PDEs from observations but rely on constructing intermediate quantities, whose errors may undermine identification accuracy and robustness. Recent advances have further incorporated deep learning into inverse PDE problems (Both et al., 2021; Chen et al., 2021; Stephany & Earls, 2024; Xu et al., 2020; Chen et al., 2022; Du et al., 2024; Gao et al., 2025; Xu et al., 2025), which exploit expressive formulations to explore equation structures but typically require system-specific training or iterative search, introducing repeated computational cost and sensitivity to modeling choices. These limitations raise the central question: Whether and to what extent could an end-to-end learning framework reconstruct the structure and coefficients of a PDE system from observations alone?

This work takes one step in this direction. We propose an end-to-end framework O_2PDE that solves inverse PDE problems from observations alone without any domain expert priors. Figure 1 provides an overview of O_2PDE, which comprises four modules: (1) a Data Encoder that converts numerical observations into latent representations, (2) a Function Encoder that transforms candidate functions as queries, (3) a Decoder with Cross-Attention that conditions queries on observation representation, and (4) a Dual Head that jointly predicts structure and coefficient identification. The whole framework is supervised by a joint classification and regression objective, together with temporal-chunk fusion and consistency regularization. Compared with existing approaches, O_2PDE maintains the end-to-end reconstruction directly from observations, not relying on domain expert priors or numerical intermediate quantities, achieving efficient inference without system-specific optimization, and equipping considerably fewer parameters, of which O_2PDE is only 5.1% of those of fine-tuned LLMs in practice. We also build an inverse-PDE-problem dataset that comprises 1.65 million PDE systems and 458.11 GB corresponding observable data. Experiments against six baselines demonstrate the effectiveness of the proposed O_2PDE. Relative to the strongest baseline for each metric, O_2PDE improves F1 by 18.4% and EM by 29.2% while reducing L2RE by 76.3% on clean observations. Even under 5% and 10% Gaussian noise, O_2PDE maintains the robustness, improving F1 by 21.7% and 18.1% and reducing L2RE by 56.4% and 47.1%, respectively.

The rest of this paper is organized as follows. Section 2 reviews related studies. Section 3 introduces the O_2PDE framework. Section 4 conducts experiments to verify the effectiveness of O_2PDE. Section 5 concludes the paper with discussions.

2 Related Work

Past decades have witnessed a lot of efforts on solving inverse PDE problems. A prominent category of solvers is traditional system identification, such as frequency-domain methods (Lataire et al., 2012; de Rozario & Oomen, 2018), set-based methods (Kuntsevich, 1996), block least mean square (Li et al., 2010), subspace methods (Verhaegen & Yu, 1995), best linear time-varying approximation (Hallemans et al., 2021; 2022), identification of time-varying systems (Chen et al., 2020), reconstruction with partial observations (Ayed et al., 2019), and compressed sensing approaches for time-varying systems (Ziniel & Schniter, 2013; Kafashan et al., 2016). The advantages of these approaches lie in their rigorous mathematical theories and high interpretability. However, this mathematical foundation imposes strong dependence on prior assumptions and domain expertise, which makes such methods difficult to generalize.

Symbolic regression techniques provide another way of solving inverse PDE problems. The typical pipeline first constructs regression quantities and then identifies the active terms and their coefficients. For example, PDE-FIND (Rudy et al., 2017) estimates numerical derivatives and reconstructs the PDEs through sparse regression. Subsequent studies improve the reliability of these intermediate quantities using robust differentiation and denoising (Zhang & Lin, 2018; He et al., 2022), neural surrogates with automatic differentiation (Xu et al., 2021), or weak formulations (Wang et al., 2019; Gurevich et al., 2019; Messenger & Bortz, 2021). The advantages of these approaches lie in the data-driven formulation and ability to recover closed-form PDEs from observations, yet they still depend heavily on accurately constructing intermediate quantities or domain-dependent expertise carefully designed since errors may still propagate to the subsequent regression.

In recent years, several solvers incorporated deep learning. DeepMoD (Both et al., 2021) combines a neural surrogate with sparse regularization. PINN-SR (Chen et al., 2021) integrates a

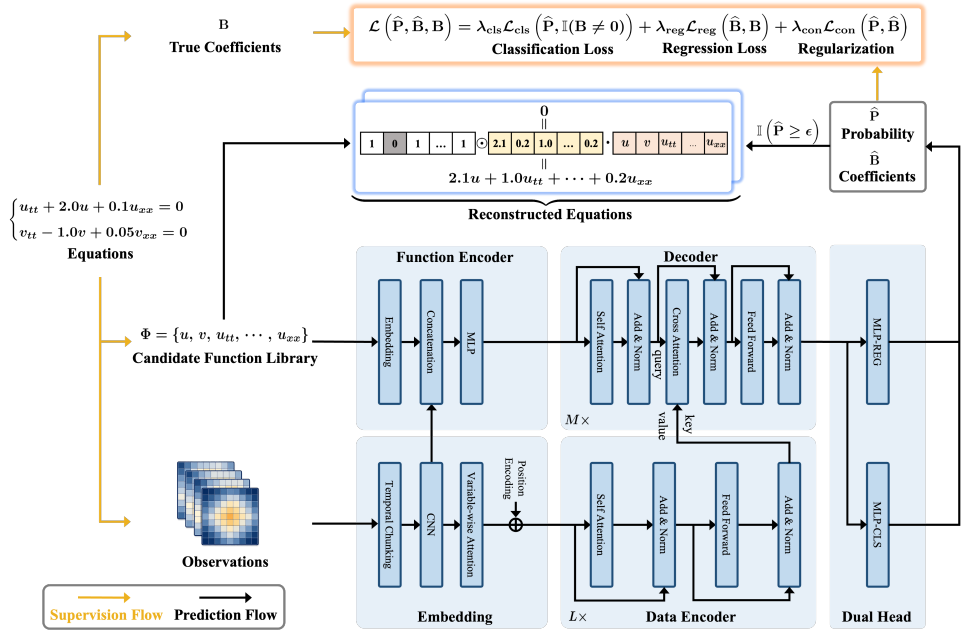


Figure 1: The workflow of the O$_2$PDE framework.

physics-informed neural network and sparse coefficient estimation. PDE-LEARN (Stephany & Earls, 2024) jointly optimizes a neural solution representation and a sparse PDE coefficient vector. DLGA-PDE (Xu et al., 2020) and SGA-PDE (Chen et al., 2022) employ evolutionary operations to search over PDE structures, whereas DISCOVER (Du et al., 2024) uses reinforcement learning to guide symbolic search. ABL-PDE (Gao et al., 2025) combines neural derivative estimation with first-order-logic reasoning to guide symbolic PDE construction through abductive learning, while EqGPT (Xu et al., 2025) learns a generative prior from PDEs collected from mathematical handbooks. The benefits of these approaches lie in the representation capacity and flexibility of deep learning models in exploring complex equation structures. Nevertheless, they generally require system-specific training, such as iterative search using method-specific priors and hyperparameter selection. Hence, reconstructing each new PDE system becomes a separate optimization problem, causing repeated computational costs and sensitivity to modeling choices.

3 The O$_2$PDE Framework

This section presents O$_2$PDE for solving inverse PDE problems from observations alone. We first formulate this task as an end-to-end prediction problem in Subsection 3.1. As illustrated in Figure 1, O$_2$PDE comprises four modules: Data Encoder that converts numerical observations into latent representations, Function Encoder that transforms candidate functions as queries, Decoder with Cross-Attention that conditions queries on observation representations, and Dual Head that jointly predicts structure and coefficient identifications, in Subsections 3.2 and 3.3. The whole framework is supervised by a joint classification and regression objective, together with temporal-chunk fusion and consistency regularization in Subsection 3.4.

3.1 Formulation of Solving Inverse PDE Problems from Observations

In this paper, we consider the PDE system of the form

$$\mathcal{F}_c\left(\mathbf{U}(\mathbf{x},t)\right)=0\,,\quad \text{for}\quad c=1,\ldots,C\,, \tag{1}$$

where $\mathbf{U}(\mathbf{x},t)=[u_1(\mathbf{x},t),\ldots,u_C(\mathbf{x},t)]$ denotes the C variables and the $\mathcal{F}_c$ is a differential operator acting on $\mathbf{U}(\mathbf{x},t)$. The $\mathbf{x}\in\mathbb{R}^D$ and $t\in\mathbb{R}$ are the spatial and temporal coordinates, respectively. Increasing D and C generally lead to more complex system dynamics and more intricate interactions

among variables. Since two-dimensional PDE systems involving two to four variables are common in physical modeling, we focus on PDE systems with $D = 2$ and $C \in \{2, 3, 4\}$. The observations are obtained by observing the PDE system on a spatial grid $\mathcal{G} = \{\mathbf{x}_{i,j}\}_{i=1,j=1}^{H,W} \subset \mathbb{R}^2$ at time steps $\mathcal{T} = \{t_n\}_{n=1}^{T} \subset \mathbb{R}$. For each variable u_c, its observations are denoted by $\mathbf{X}_c \in \mathbb{R}^{H\times W\times T}$, where $(\mathbf{X}_c)_{i,j,n} = u_c(\mathbf{x}_{i,j}, t_n)$. The observations of the PDE system are then represented as $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_C] \in \mathbb{R}^{C\times H\times W\times T}$. It is noticed that each equation can be expressed as a linear combination of functions involving the variables and their derivatives. For instance, one has

$$\begin{cases} u_{tt} + 2.0u + 0.1u_{xx} = 0\,, \\ v_{tt} - 1.0v + 0.05v_{xx} = 0\,. \end{cases} \quad \Rightarrow \quad \underbrace{\begin{bmatrix} 1.0 & 2.0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & -1.0 & 0.05 \end{bmatrix}}_{\text{refers to coefficients}} \underbrace{\begin{bmatrix} u_{tt} & u & u_{xx} & v_{tt} & v & v_{xx} \end{bmatrix}^\top}_{\text{refers to structures}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\,.$$

Accordingly, solving the inverse PDE problem can be equivalently converted into the problem of structure identification and coefficient identification, where the former determines which functions are present in the PDE system, while the latter estimates the corresponding coefficients.

The candidate function library $\mathbf{\Phi} = \{\phi_1, \ldots, \phi_N\}$ is constructed empirically by collecting all the functions that might appear in the PDE system. For convenience, we reuse $\mathbf{\Phi}$ to denote the corresponding vector of symbolic candidate function library $[\phi_1, \ldots, \phi_N]^\top$, leading to an alternative form $\mathbf{B\Phi} = \mathbf{0}$ of the PDE system in Eq. (1), where $\mathbf{B} = [B_{c,j}]_{c,j} \in \mathbb{R}^{C\times N}$ is the coefficient matrix and $B_{c,j}$ is the coefficient of candidate function ϕ_j in the c-th equation. The $\mathbb{I}(\mathbf{B} \neq 0) \in \{0, 1\}^{C\times N}$ is the structure matrix, specifying which functions are active in the PDE system. The $\mathbb{I}(\mathbf{B} \neq 0)$ and $\mathbf{B}$ are the objectives of the structure identification and coefficient identification, respectively. Thus, solving the inverse PDE problem from observations alone is formulated as learning an end-to-end model f_θ that jointly predicts the structure and coefficient matrices from observations as $(\hat{\mathbf{P}}, \hat{\mathbf{B}}) = f_\theta(\mathbf{X})$, where $\hat{\mathbf{P}} \in [0, 1]^{C\times N}$ contains the predicted probabilities that candidate functions are active and $\hat{\mathbf{B}} \in \mathbb{R}^{C\times N}$ contains the predicted coefficients. Therefore, the reconstructed PDE system is given by $\mathbb{I}(\hat{\mathbf{P}} \geq \epsilon) \odot \hat{\mathbf{B}}\mathbf{\Phi} = \mathbf{0}$, where $\odot$ denotes the Hadamard product and $\epsilon \geq 0$ is the predefined threshold.

3.2 Data and Function Encoder

The Data Encoder converts the observations into latent representations and the Function Encoder transforms symbolic candidate functions into queries.

Embedding. We first embed each variable channel of observations $\mathbf{X}_c$ by a shared CNN. Let

$$\mathbf{Z}_c = \mathrm{CNN}_{\theta_{\mathrm{cnn}}}(\mathbf{X}_c) \in \mathbb{R}^{N_x\times d} \tag{2}$$

denote the resulting N_x d-dimensional features for variable u_c. Since PDE systems depend on interactions among variables, we apply variable-wise attention across variables at each corresponding position. We then augment the results with positional encodings $\mathbf{Pos} \in \mathbb{R}^{CN_x\times d}$ as

$$\mathbf{E} = \mathrm{Attn}_{\mathrm{var}}\left(\mathbf{Z}_1, \ldots, \mathbf{Z}_C\right) + \mathbf{Pos} \in \mathbb{R}^{CN_x\times d}\,.$$

Data Encoder. We then process the augmented representations using an L-layer Transformer encoder (Vaswani et al., 2017) to capture global dependencies. The results are the latent representations of observations, which are denoted by $\mathbf{H} = \mathrm{Enc}_{\theta_{\mathrm{enc}}}(\mathbf{E}) \in \mathbb{R}^{CN_x\times d}$.

Function Encoder. Although candidate functions may take diverse symbolic forms, each function can be decomposed into some shared factors. Exploiting this property, we first decompose each candidate function into its constituent factors. Specifically, let

$$\phi_j = \prod_{k=1}^{K_j} f_{j,k} \quad \text{and} \quad f_{j,k} = \mathcal{D}_{j,k} u_{a_{j,k}}\,,$$

where $a_{j,k} \in \{1, \ldots, C\}$ denotes the index of the variable in the k-th factor of ϕ_j. Each factor $f_{j,k}$ pairs the variable $u_{a_{j,k}}$ with an operator $\mathcal{D}_{j,k}$. For example, $\partial u_1/\partial x$ pairs u_1 with $\partial/\partial x$. Then the factors are embedded individually and aggregated into the candidate function representations. Specifically, the variable representation $\mathbf{s}_{u_{a_{j,k}}}$ is obtained by mean-pooling $\mathbf{Z}_{a_{j,k}}$ in Eq. (2) over the

N_x dimension, while the operator $\mathcal{D}_{j,k}$ is represented by a learnable embedding. We then concatenate and pass them through an MLP G_{θ_e} to obtain the factor embedding as

$$\mathbf{e}^f_{j,k} = G_{\theta_e}\left([\mathbf{s}_{u_{a_{j,k}}} \,;\, \mathrm{Emb}(\mathcal{D}_{j,k})]\right) \in \mathbb{R}^d \,.$$

The candidate function representation is then obtained by aggregating its factor embeddings as $\mathbf{e}^\phi_j = \sum_{k=1}^{K_j} \mathbf{e}^f_{j,k} \in \mathbb{R}^d$. Particularly, the constant candidate function 1 is forced as a learnable embedding. Since the same candidate function may appear in equations governing different variables, we condition $\mathbf{e}^\phi_j$ on the representation $\mathbf{s}_{u_c}$ of the target variable u_c by an MLP G_{θ_q} as $\mathbf{q}_{c,j} = G_{\theta_q}([\mathbf{e}^\phi_j \,;\, \mathbf{s}_{u_c}]) \in \mathbb{R}^d$. Applying this construction to the candidate function library yields the queries

$$\mathbf{Q} = [\mathbf{q}_{1,1}, \ldots, \mathbf{q}_{1,N}, \ldots, \mathbf{q}_{C,1}, \ldots, \mathbf{q}_{C,N}]^\top \in \mathbb{R}^{CN\times d} \,.$$

3.3 Decoder and Dual Head

The Decoder with Cross-Attention conditions the queries $\mathbf{Q}$ on the observation representation $\mathbf{H}$. The resulting features are then passed to the Dual Head, which jointly predicts the PDE structure and coefficient identifications.

Decoder with Cross-Attention. Specifically, we employ an M-layer Transformer decoder. Each decoder layer first applies self-attention among the queries. Cross-attention is then performed using the resulting query representations as queries and $\mathbf{H}$ as keys and values, followed by a feed-forward network. The resulting features are denoted by $\mathbf{R} = \mathrm{Dec}_{\theta_{\mathrm{dec}}}(\mathbf{Q}, \mathbf{H}) \in \mathbb{R}^{CN\times d}$.

Dual Head. Each $\mathbf{r}_{c,j} \in \mathbb{R}^d$ in $\mathbf{R}$ represents candidate function ϕ_j conditioned on the observation for the c-th equation. Thus, we add classification and regression heads to $\mathbf{r}_{c,j}$, so that prediction of presence probability and coefficient of candidate function ϕ_j in the c-th equation becomes

$$\hat{P}_{c,j} = \sigma\left(G_{\theta_{\mathrm{cls}}}(\mathbf{r}_{c,j})\right) \quad \text{and} \quad \hat{B}_{c,j} = G_{\theta_{\mathrm{reg}}}(\mathbf{r}_{c,j}) \,,$$

where $G_{\theta_{\mathrm{cls}}}$ and $G_{\theta_{\mathrm{reg}}}$ separately denote the classification and regression heads, and $\sigma(\cdot)$ is the sigmoid function. Stacking the predictions yields the probability and coefficient matrices as

$$\hat{\mathbf{P}} = \left[\hat{P}_{c,j}\right]_{c,j} \in [0,1]^{C\times N} \quad \text{and} \quad \hat{\mathbf{B}} = \left[\hat{B}_{c,j}\right]_{c,j} \in \mathbb{R}^{C\times N} \,.$$

3.4 Training and Regularization

This subsection introduces the joint classification and regression objective for training the whole model, together with temporal-chunk fusion and consistency regularization. We first segment observations into K temporal chunks of equal length t as $\mathcal{X} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(K)}\}$, where $\mathbf{X}^{(k)} \in \mathbb{R}^{C\times H\times W\times t}$. The consecutive chunks may overlap along the temporal dimension. Each chunk is processed independently according to $(\hat{\mathbf{P}}^{(k)}, \hat{\mathbf{B}}^{(k)}) = f_\theta(\mathbf{X}^{(k)})$ for $k = 1, \ldots, K$. The chunk-level averages will serve as the final predictions, which are denoted by $\overline{\mathbf{P}} = \sum_{k=1}^K \hat{\mathbf{P}}^{(k)}/K$ and $\overline{\mathbf{B}} = \sum_{k=1}^K \hat{\mathbf{B}}^{(k)}/K$. The average predictions $\overline{\mathbf{P}}$ and $\overline{\mathbf{B}}$ are supervised by the structure matrix $\mathbb{I}(\mathbf{B} \neq 0)$ and the coefficient matrix $\mathbf{B}$, respectively.

For notational convenience, we reuse $\hat{\mathbf{P}}$ and $\hat{\mathbf{B}}$ to denote the collections of chunk-level predictions $\{\hat{\mathbf{P}}^{(k)}\}_{k=1}^K$ and $\{\hat{\mathbf{B}}^{(k)}\}_{k=1}^K$, respectively. For structure identification, the target $\mathbb{I}(\mathbf{B} \neq 0)$ is a binary matrix indicating whether each candidate function is active in the PDE system. We formulate it as a multi-label classification task. The candidate function library typically contains far more terms than are active in each equation, leading to an imbalance between positive and negative targets. To mitigate this imbalance, we use the asymmetric loss (Ridnik et al., 2021). It applies class-dependent focusing and probability shifting to suppress the contribution of easy negatives while retaining informative positive and hard negative signals. For coefficient identification, the target $\mathbf{B}$ is a real-valued matrix specifying the coefficient of each candidate function. We formulate it as a regression task. To reduce the influence of large residuals while preserving sensitivity to small residuals, we use the Huber loss (Huber, 1964). It is quadratic for small residuals and grows linearly beyond a threshold, enabling robust yet precise regression. The two losses are denoted by

$$\mathcal{L}_{\mathrm{cls}}\left(\hat{\mathbf{P}}, \mathbb{I}(\mathbf{B} \neq 0)\right) = \ell_{\mathrm{ASL}}\left(\overline{\mathbf{P}}, \mathbb{I}(\mathbf{B} \neq 0)\right) \quad \text{and} \quad \mathcal{L}_{\mathrm{reg}}\left(\hat{\mathbf{B}}, \mathbf{B}\right) = \ell_{\mathrm{Huber}}\left(\overline{\mathbf{B}}, \mathbf{B}\right) \,.$$

Since all temporal chunks correspond to the same PDE system, the predictions should remain invariant across chunks. We introduce a consistency loss that penalizes their chunk-wise variances. Specifically, we compute the variances of the predictions across chunks as

$$\mathcal{L}_{\text{con}}\left(\hat{\mathbf{P}}, \hat{\mathbf{B}}\right) = \frac{1}{KCN} \sum_{k=1}^{K} \left[\left\| \hat{\mathbf{P}}^{(k)} - \overline{\mathbf{P}} \right\|_F^2 + \gamma \left\| \hat{\mathbf{B}}^{(k)} - \overline{\mathbf{B}} \right\|_F^2 \right] ,$$

where $\| \cdot \|_F$ denotes the Frobenius norm and $\gamma \geq 0$ balances the variances. Combining the three components gives the training objective as

$$\mathcal{L}\left(\hat{\mathbf{P}}, \hat{\mathbf{B}}, \mathbf{B}\right) = \lambda_{\text{cls}} \mathcal{L}_{\text{cls}}\left(\hat{\mathbf{P}}, \mathbb{I}(\mathbf{B} \neq 0)\right) + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}\left(\hat{\mathbf{B}}, \mathbf{B}\right) + \lambda_{\text{con}} \mathcal{L}_{\text{con}}\left(\hat{\mathbf{P}}, \hat{\mathbf{B}}\right) , \tag{3}$$

where $\lambda_{\text{cls}} \geq 0$, $\lambda_{\text{reg}} \geq 0$, and $\lambda_{\text{con}} \geq 0$. During inference, we apply the same temporal chunking procedure. We then threshold the averaged probabilities using $\epsilon \geq 0$ and retain the averaged coefficients, yielding the reconstructed PDE system $\mathbb{I}(\overline{\mathbf{P}} \geq \epsilon) \odot \overline{\mathbf{B}}\mathbf{\Phi} = \mathbf{0}$.

4 Experiments

This section conducts experiments to evaluate the effectiveness and robustness of O_2PDE. We comprehensively examine the reconstruction performance of the proposed O_2PDE and its contenders within noisy observations alone.

Table 1: Descriptions of the conducted datasets.

Number of variables	Samples	Size	Library size	Active terms per equation
2	0.55 M	101.69 GB	26	10.18
3	0.55 M	152.66 GB	46	10.53
4	0.55 M	203.76 GB	71	10.73

4.1 Experimental Configurations

Datasets. For the inverse PDE problem solving, we consider the typical PDEs with $C = 2$, 3, and 4 variables, originating from the 2D wave systems (Ye et al., 2025). For each variable count C, the corresponding dataset contains 0.55 million instances, giving 1.65 million instances in total. Each instance consists of a PDE system and its corresponding observations, where the former is regarded as the underlying target, whereas the latter is used as observations alone. We adapt the generation protocol of PDEformer-2 (Ye et al., 2025) with the modifications summarized in Appendix A. The equation structures combine linear, nonlinear, derivative, and cross-variable terms, with the corresponding candidate function libraries increasing in size as C grows. For each PDE system, we employ the conventional PDE solver (Burns et al., 2020) to obtain observations on a 32×32 spatial grid at 26 time steps. Thus, we have an underlying PDE system and a collection of observations with $32 \times 32 \times 26$ dimensions in each instance. Table 1 summarizes the conducted datasets, categorized by the number of PDE variables. To evaluate robustness to observational noise, we also add Gaussian noise at levels of 5% and 10%, following the configurations of Gao et al. (2025). Appendix A provides the further details of the dataset generation and noise protocol.

Baselines. Here, we compare O_2PDE with representative inverse PDE problem methods, including PDE-FIND (Rudy et al., 2017), DL-PDE (Xu et al., 2021), WSINDy (Messenger & Bortz, 2021), SGA-PDE (Chen et al., 2022), and ABL-PDE (Gao et al., 2025). Note that we also fine-tune Qwen3-VL-2B-Instruct (Bai et al., 2025) on the training dataset as a generic end-to-end baseline. Detailed baseline configurations can be obtained in Appendix B.

Evaluation metrics. We evaluate the inverse PDE problem from two aspects, that is, structure identification and coefficient identification, where the former determines which candidate functions are active in each equation, while the latter estimates their corresponding numerical coefficients (Luenberger, 1979). We employ the F1 metric with micro aggregation and the Exact Match (EM) metric for examination of structure identification, while the coefficient identification is examined by the L2RE metric over the union of nonzero ground-truth and predicted terms. Table 8 summarizes these metrics, and detailed information can be accessed in Appendix C.

Table 2: Performance of O_2PDE and six contenders under clean and noisy observations.

Methods	0% noise			5% noise			10% noise		
	F1 ↑	EM ↑	L2RE ↓	F1 ↑	EM ↑	L2RE ↓	F1 ↑	EM ↑	L2RE ↓
PDE-FIND	0.6797	3.12	3.3853	0.4876	0.10	11.6952	0.4545	0.00	14.5272
DL-PDE	0.6657	3.67	3.1576	0.5699	1.11	5.7617	0.5103	**0.56**	7.9113
WSINDy	0.8122	11.73	1.4596	0.3574	0.16	9.1582	0.3132	0.01	14.2981
SGA-PDE	–	23.22	0.3180	–	**1.56**	3.1338	–	0.00	6.0698
ABL-PDE	0.5761	1.56	8.0459	0.5531	1.44	9.0674	0.5142	0.44	10.7889
LLM-SFT (2B)	0.5012	0.00	1.4411	0.5011	0.00	1.4507	0.5006	0.00	1.4664
O_2PDE (101.81M)	**0.9618**	**52.42**	**0.0754**	**0.6934**	1.31	**0.6328**	**0.6075**	0.24	**0.7760**

Table 3: Performance of O_2PDE relative to the number of variables.

C	0% noise			5% noise			10% noise		
	F1 ↑	EM ↑	L2RE ↓	F1 ↑	EM ↑	L2RE ↓	F1 ↑	EM ↑	L2RE ↓
2	0.9691	58.27	0.0617	0.7592	2.11	0.5543	0.6968	0.40	0.7168
3	0.9644	54.35	0.0693	0.7007	1.33	0.6228	0.6117	0.24	0.7721
4	0.9563	48.04	0.0869	0.6515	0.89	0.6796	0.5544	0.16	0.8085

4.2 Reconstruction Performance

This subsection evaluates the reconstruction performance and robustness of O_2PDE against variables and noises. We first compare O_2PDE with six contenders under 0%, 5%, and 10% observational noise. The empirical results are listed in Table 2, from which O_2PDE achieves the best performance on clean observations, evaluated by all three metrics, where O_2PDE improves 18.4% F1 and 29.2% EM and reduces 76.3% L2RE than the SOTA contender. Even corrupting 5% and 10% Gaussian noises to observations, O_2PDE separately improves F1 by 21.7% and 18.1%, reduces L2RE by 56.4% and 47.1%, and obtains comparable EM in comparison with the SOTA baseline. Especially with 101.81M parameters that are approximately 5.1% of LLM's, O_2PDE achieves 91.9% F1 and 52.42% EM improvements and reduces 94.8% L2RE rather than those of fine-tuned LLMs. These observations confirm the reconstruction effectiveness and the robustness of O_2PDE against noises.

Next, we further investigate the reconstruction performance of O_2PDE relative to the number of system variables C, where the reconstruction task as well as the PDE system becomes very complex as the number of variables increases. Table 3 lists the variable-wise reconstruction performance of O_2PDE. It is observed that as C increases from 2 to 4, F1 decreases by 1.32%, 14.19%, and 20.44%, EM drops by 10.23%, 1.22%, and 0.24%, and L2RE increases by 40.84%, 22.61%, and 12.79% under 0%, 5%, and 10% noise, respectively.

The aforementioned metrics work for evaluating the overall reconstruction performance but mask term-specific behavior of PDE reconstruction. Hence, we further introduce term-level metrics to identify which candidate function families are reliably recovered, where term-level F1 and L2RE are separately computed only based on the occurrences of that term without averaging and the term coefficients. Figure 2 displays the term-level F1 and L2RE of O_2PDE across candidate function families on clean observations, where three panels show PDE systems with C variables, where each spoke represents a candidate function, braces denote the function families that are classified as derivative-related or derivative-free based on cases that they involve derivatives or not, the derivative subscripts specify the variables and differential order, such as $u_{jx} = \partial u_j/\partial x$ and $u_{jyy} = \partial^2 u_j/\partial y^2$. It is observed that O_2PDE reliably identifies derivative-related terms, achieving F1 of at least 0.95, while maintaining L2RE below 0.07. In contrast, accurately identifying derivative-free terms remains more challenging. We conjecture that correlations among derivative-free terms make them harder to distinguish, whereas CNNs' inductive bias toward local features is better suited to derivative-related terms, which capture local spatiotemporal variations.

4.3 Scalability

This subsection investigates the scalability of our proposed O_2PDE, i.e., whether and to what extent scaling up model capacity would affect the reconstruction performance of O_2PDE. Here, we focus

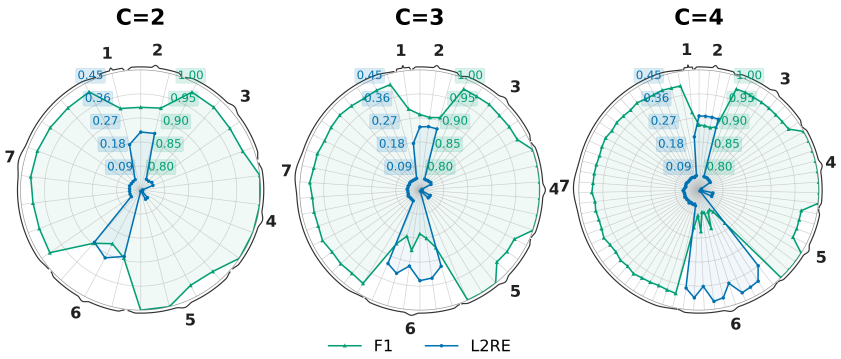


Figure 2: Radar plots of the term-level F1 and L2RE of O_2PDE on clean observations. Here, the derivative-free one contains **1** Constant (1), **2** Linear value (u_j), and **6** Symmetric quadratic value ($u_j u_k, j \leq k$), while the derivative-related one comprises **3** First spatial derivative (u_{jx}, u_{jy}), **4** Second spatial derivative (u_{jxx}, u_{jyy}), **5** Temporal derivative (u_{jt}, u_{jtt}), and **7** Value-gradient product ($u_j u_{kx}, u_j u_{ky}$).

Table 4: Performance of O_2PDE with respect to increasing the hidden dimension h of embedding CNN, the encoder depth L, and the decoder depth M.

h	L	M	Params. (M)	F1 ↑	EM ↑	L2RE ↓	Computational cost (GFLOPs/sample)
128	8	8	24.39	0.9462	40.94	0.1112	386.7
128	12	12	31.77	0.9536	45.85	0.0947	399.8
128	24	24	53.89	0.9576	48.81	0.0853	439.2
128	50	50	101.81	**0.9618**	**52.42**	**0.0754**	524.5
256	12	12	44.32	0.9563	47.50	0.0889	1424.1

on three key factors, including the hidden dimension h of embedding CNN, the encoder depth L, and the decoder depth M.

Table 4 lists the reconstruction performance of O_2PDE with respect to increasing values of h, L, and M. It is observed that when h is fixed at 128, increasing $L = M$ from 8 to 50 improves F1 and EM by 1.6% and 11.5%, respectively, while reducing L2RE by 32.2%. Moreover, the model with $L = M = 24$ and $h = 128$ outperforms the model with $L = M = 12$ and $h = 256$ on all three metrics while requiring only 439.2 rather than 1424.1 GFLOPs per sample, corresponding to a 69.2% reduction.

Figure 3 plots the performance curves of O_2PDE with respect to increasing model capacity. It is observed that reconstruction performance continues to improve as the parameter count increases, with no evident saturation in either structure or coefficient identification. These observations show that O_2PDE continues to benefit from increased model capacity, while depth scaling is more compute-efficient than width scaling among the evaluated configurations, which may imply that a scaling law also exists in PDE reconstruction foundation models.

4.4 Ablation Studies

This subsection conducts ablation studies to investigate the benefits of four key components in O_2PDE, including the function encoder, consistency loss, temporal chunking, and dual head. Accordingly, we design four variants by removing or modifying each component, that is, w/o function encoder that replaces the dynamically generated queries with fixed learnable queries, w/o consistency loss that removes the consistency loss while retaining the temporal chunking, w/o temporal chunking that processes observations without the temporal chunking, and w/o dual head that uses only the regression head and identifies active terms by thresholding the predicted coefficient.

Table 5 reports the reconstruction performance of O_2PDE and its four variants, from which removing or modifying the function encoder, consistency loss, temporal chunking, and dual head separately results in the decreasing of F1 by 0.1%, 0.9%, 1.5%, and 1.6%, where the corresponding decreases

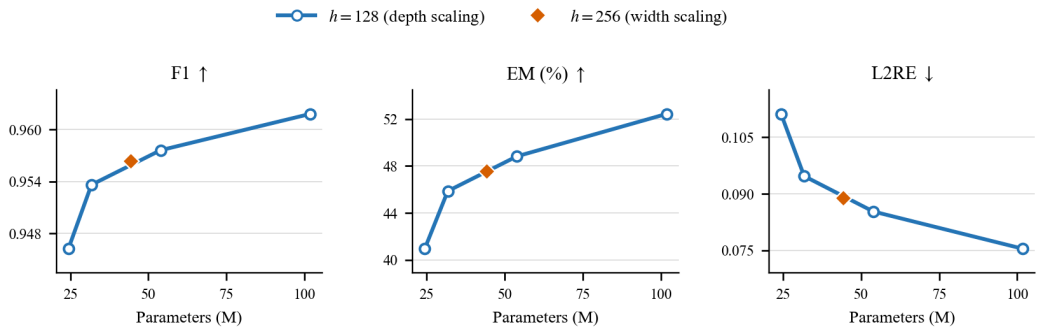


Figure 3: Performance of O_2PDE as model parameters scale up.

Model	F1 ↑	EM (%) ↑	L2RE ↓
O_2PDE	**0.9618**	**52.42**	**0.0754**
w/o function encoder	0.9609	51.96	0.0768
w/o consistency loss	0.9533	48.69	0.0860
w/o temporal chunking	0.9469	43.03	0.1030
w/o dual head	0.9461	42.67	0.1046

Table 5: Performance of O_2PDE and its variants.

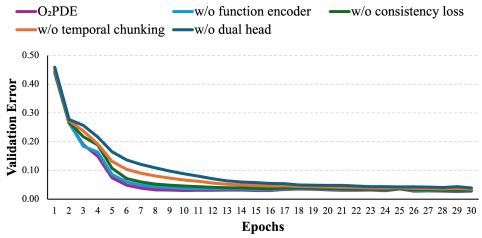


Figure 4: Validation error curves of O_2PDE and its variants.

in EM are 0.9%, 7.1%, 17.9%, and 18.6%, while the increases in L2RE are 1.9%, 14.1%, 36.6%, and 38.7%, respectively. Figure 4 plots the validation error curves of O_2PDE and its four variants over 30 training epochs, where the validation error consists of the classification and regression ones in Eq. (3). It is obvious that all models converge, whereas O_2PDE achieves the rapidest convergence rate and the lowest error. These observations confirm the effectiveness of four key components.

5 Conclusion and Discussions

In this paper, we proposed an end-to-end framework O_2PDE for inverse PDE problem solving. In contrast to conventional approaches, O_2PDE not only reconstructs PDE systems directly from observations without relying on domain expert priors or numerical intermediate quantities, but also achieves efficient inference without system-specific optimization. This work also constructed a dataset relative to the inverse PDE problem for evaluation, which contains 1.65 million PDE systems with various variables and 458.11 GB of corresponding observable data. Experimental results show that our proposed O_2PDE achieves, for the first time, the F1, EM, and L2RE exceeding 0.9618, 52.42%, and 0.0754, respectively, significantly surpassing existing SOTA baselines. This is a laudable result, confirming the reconstruction advances of O_2PDE with respect to four key components.

Despite the significant reconstruction performance achieved in this work, there are still issues worth exploring in solving inverse PDE problems from observations. First, when an active ground-truth term has a coefficient with small magnitude, its impact on the PDE system may be weak, often making it difficult to distinguish it from inactive ones. This conjecture is empirically verified in Appendix E. Second, whether it is necessary and how to effectively incorporate domain expert priors, such as symmetry or conservation constraints, into the foundation model remains challenging. These challenges motivate several research directions in the future. On the one hand, developing richer function representations, such as encoding the structures of domain expert priors, while maintaining end-to-end inference. On the other hand, exploring alternative structure identification paradigms like generative models. We expect these extensions to broaden the applicability of our proposed framework without sacrificing inference efficiency and reconstruction performance.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript translation and polishing. We have not used generative AI tools for the following tasks, including generating synthetic data sets, developing theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, and interpreting results. The following tasks are not applicable to this work: Formulate mathematical claims; provide critical ingredients for proving mathematical claims; assist in the writing of proofs; propose or refine hypotheses; support qualitative and thematic data analysis. We have reviewed all AI-assisted work; even the AI-assisted translation was checked by the authors to ensure that it preserved the intended meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Uri M Ascher, Steven J Ruuth, and Brian T R Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari. Learning dynamical systems from partial observations. *arXiv preprint arXiv:1902.11136*, 2019.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Jens Berg and Kaj Nyström. Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, 384:239–252, 2019.

Gert-Jan Both, Subham Choudhury, Pierre Sens, and Remy Kusters. DeepMoD: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, 428:109985, 2021.

Keaton J Burns, Geoffrey M Vasil, Jeffrey S Oishi, Daniel Lecoanet, and Benjamin P Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2(2):023068, 2020.

Siyi Chen, Jubin Lu, and Ying Lei. Identification of time-varying systems with partial acceleration measurements by synthesis of wavelet decomposition and Kalman filter. *Advances in Mechanical Engineering*, 12(6):1687814020930460, 2020.

Yuntian Chen, Yingtao Luo, Qiang Liu, Hao Xu, and Dongxiao Zhang. Symbolic genetic algorithm for discovering open-form partial differential equations (SGA-PDE). *Physical Review Research*, 4(2):023174, 2022.

Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12(1):6136, 2021.

Robin de Rozario and Tom Oomen. Frequency response function identification of LPV systems: A global approach with application to mechanical systems. *IFAC-PapersOnLine*, 51(15):108–113, 2018.

Mengge Du, Yuntian Chen, and Dongxiao Zhang. DISCOVER: Deep identification of symbolically concise open-form partial differential equations via enhanced reinforcement learning. *Physical Review Research*, 6(1):013182, 2024.

Enhao Gao, Cunjing Ge, Yuan Jiang, and Zhihua Zhou. Discovering symbolic partial differential equation by abductive learning. In *Advances in Neural Information Processing Systems 38*, pp. 57359–57385, 2025.

Daniel R Gurevich, Patrick AK Reinbold, and Roman O Grigoriev. Robust and optimal sparse regression for nonlinear PDE models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10), 2019.

Noël Hallemans, Rik Pintelon, Els Van Gheem, Thomas Collet, Raf Claessens, Benny Wouters, Kristof Ramharter, Annick Hubin, and John Lataire. Best linear time-varying approximation of a general class of nonlinear time-varying systems. *IEEE Transactions on Instrumentation and Measurement*, 70:1–14, 2021.

Noël Hallemans, Rik Pintelon, Xinhua Zhu, Thomas Collet, Meisam Dabiri Havigh, Benny Wouters, Reynier I Revilla, Raf Claessens, Kristof Ramharter, Annick Hubin, and John Lataire. Trend removal in measurements of best linear time-varying approximations—with application to operando electrochemical impedance spectroscopy. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022.

Yuchen He, Sung-Ha Kang, Wenjing Liao, Hao Liu, and Yingjie Liu. Robust identification of differential equations by numerical techniques from a single set of noisy observation. *SIAM Journal on Scientific Computing*, 44(3):A1145–A1175, 2022.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Chris Huntingford, Andrew J. Nicoll, Cornelia Klein, and Jawairia A. Ahmad. Potential for equation discovery with AI in the climate sciences. *Earth System Dynamics*, 16(2):475–495, 2025.

MohammadMehdi Kafashan, Anirban Nandi, and ShiNung Ching. Relating observability and compressed sensing of time-varying signals in recurrent linear networks. *Neural Networks*, 83:11–20, 2016.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Vsevolod M Kuntsevich. Set-valued estimation of state and parameter vectors within adaptive control systems. In *Bounding Approaches to System Identification*, pp. 239–259, 1996.

John Lataire, Rik Pintelon, and Ebrahim Louarroudi. Non-parametric estimate of the system function of a time-varying system. *Automatica*, 48(4):666–672, 2012.

Yang Li, Hualiang Wei, and Stephen A Billings. Identification of time-varying systems using multiwavelet basis functions. *IEEE Transactions on Control Systems Technology*, 19(3):656–663, 2010.

David G Luenberger. *Introduction to dynamic systems: Theory, models, and applications*. John Wiley & Sons, 1979.

Daniel A Messenger and David M Bortz. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021.

Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *2021 IEEE/CVF International Conference on Computer Vision*, pp. 82–91, 2021.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

Siddhartha Srivastava, Patrick C Kinnunen, Zhenlin Wang, Kenneth K Y Ho, Brock A Humphries, Siyi Chen, Jennifer J Linderman, Gary D Luker, Kathryn E Luker, and Krishna Garikipati. Inference of weak-form partial differential equations describing migration and proliferation mechanisms in wound healing experiments on cancer cells. *PLOS Computational Biology*, 21(10): e1013607, 2025.

Robert Stephany and Christopher Earls. PDE-LEARN: Using deep learning to discover partial differential equations from noisy, limited data. *Neural Networks*, 174:106242, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017.

Michel Verhaegen and Xiaode Yu. A class of subspace model identification algorithms to identify periodically and arbitrarily time-varying systems. *Automatica*, 31(2):201–216, 1995.

Zhenlin Wang, Xun Huan, and Krishna Garikipati. Variational system identification of the partial differential equations governing the physics of pattern-formation: Inference under varying fidelity and noise. *Computer Methods in Applied Mechanics and Engineering*, 356:44–74, 2019.

Hao Xu, Haibin Chang, and Dongxiao Zhang. DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm. *Journal of Computational Physics*, 418:109584, 2020.

Hao Xu, Haibin Chang, and Dongxiao Zhang. DL-PDE: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data. *Communications in Computational Physics*, 29(3):698–728, 2021.

Hao Xu, Yuntian Chen, Rui Cao, Tianning Tang, Mengge Du, Jian Li, Adrian H. Callaghan, and Dongxiao Zhang. Generative discovery of partial differential equations by learning from math handbooks. *Nature Communications*, 16(1):10255, 2025.

Zhanhong Ye, Zining Liu, Bingyang Wu, Hongjie Jiang, Leheng Chen, Minyan Zhang, Xiang Huang, Qinghe Meng, Jingyuan Zou, Hongsheng Liu, and Bin Dong. PDEformer-2: A versatile foundation model for two-dimensional partial differential equations. *arXiv preprint arXiv:2507.15409*, 2025.

Laure Zanna and Thomas Bolton. Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, 47(17):e2020GL088376, 2020.

Sheng Zhang and Guang Lin. Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474 (2217):20180305, 2018.

Justin Ziniel and Philip Schniter. Dynamic compressive sensing of time-varying signals via approximate message passing. *IEEE Transactions on Signal Processing*, 61(21):5270–5284, 2013.

A Dataset Construction

This section describes the generation of the dataset, the candidate function library, the dataset splits, and the observational noise protocol used in the experiments.

A.1 Dataset Generation

This subsection describes how the PDE systems and their observations are generated. We adapt the MV-Wave generator of PDEformer-2 (Ye et al., 2025). For each variable u_i, the generated system has the form

$$\partial_{tt}u_i + \mu_i\partial_t u_i - \kappa_i\Delta u_i + f_0(\mathbf{u})_i + s_i + \partial_x f_1(\mathbf{u})_i + \partial_y f_2(\mathbf{u})_i = 0\ ,$$

where $\mathbf{u} = (u_1, \ldots, u_C)$. For $l \in \{0, 1, 2\}$, the interaction function retains the sparse linear–quadratic form

$$f_l(\mathbf{u})_i = \sum_{j=1}^{C} a_{lij}u_j + \sum_{1\le j\le k\le C} b_{lijk}u_j u_k\ .$$

For each f_l, at most $3C$ terms are active; each active coefficient equals 1 with probability 0.5 and otherwise follows $\mathcal{U}(-1, 1)$. The initial-condition distribution, periodic unit-square domain, and trajectory rejection rule follow the original protocol. Our generator differs from the original in three respects. First, the damping and source terms are restricted to spatial constants sampled from $\{0, 1, \mathcal{U}(-1, 1)\}$, and the spatial second-order term is fixed to $-\kappa_i\Delta u_i$. Here, κ_i is sampled log-uniformly from $[0.01, 4]$ and is set to zero with probability 0.2 only when neither flux function affecting the i-th equation depends on u_i; otherwise, it remains nonzero. Second, for the reconstruction labels, coefficients with magnitude below 10^{-9} are removed, and flux terms are expanded using the product rule. Third, we solve each system using Dedalus v3 (Burns et al., 2020) with SBDF2 time stepping (Ascher et al., 1995) on a 32×32 Fourier grid, using $3/2$ dealiasing and a maximum internal step size of 0.01. We record 26 equally spaced frames on $t \in [0, 1]$, including both endpoints, rather than the original 128×128 spatial grid and 101 recorded frames.

Table 6: Dataset statistics.

C	Library size	Grid	Training	Validation	Test
2	26	$32 \times 32 \times 26$	0.45 M	0.05 M	0.05 M
3	46	$32 \times 32 \times 26$	0.45 M	0.05 M	0.05 M
4	71	$32 \times 32 \times 26$	0.45 M	0.05 M	0.05 M

The training, validation, and test sets are generated using distinct random seeds and checked for duplicates, ensuring that no identical system appears in any pair of splits.

A.2 Candidate Function Library

This subsection defines the shared candidate function library. Table 7 lists the candidate function families. Here, the derivative subscripts specify the differentiation coordinate and order. For example, $u_{jx} = \partial u_j/\partial x$, $u_{jxx} = \partial^2 u_j/\partial x^2$, and $u_{jtt} = \partial^2 u_j/\partial t^2$. The indices j and k may refer to either the same variable or different variables, so the library covers both within-variable dynamics and cross-variable coupling. Summing the number of terms in all families gives the library size

$$N(C) = 1 + C + 2C + 2C + 2C + \frac{C(C+1)}{2} + 2C^2 = \frac{5C^2 + 15C + 2}{2}\ ,$$

which gives 26, 46, and 71 terms for $C = 2, 3, 4$, respectively. All library-based methods use the same candidate function library. The families can be divided into derivative-free and derivative-related terms. The derivative-free group contains the constant 1, the linear values u_j, and the symmetric quadratic products $u_j u_k$. The constant represents a state-independent source or offset; the linear values describe individual-variable effects and linear coupling; and the quadratic products represent nonlinear self-interactions when $j = k$ and cross-variable interactions when $j \neq k$. Because multiplication is commutative, retaining only $j \le k$ removes duplicate products such as $u_j u_k$

and $u_k u_j$. The derivative-related group contains first- and second-order spatial derivatives, temporal derivatives, and value–gradient products. Specifically, u_{jx} and u_{jy} describe directional first-order transport, u_{jxx} and u_{jyy} capture second-order spatial effects such as Laplacian contributions, and u_{jt} and u_{jtt} represent first- and second-order temporal dynamics. Finally, $u_j u_{kx}$ and $u_j u_{ky}$ encode nonlinear, direction-dependent coupling between a field value and a spatial gradient.

Table 7: Candidate function families for a system with C variables.

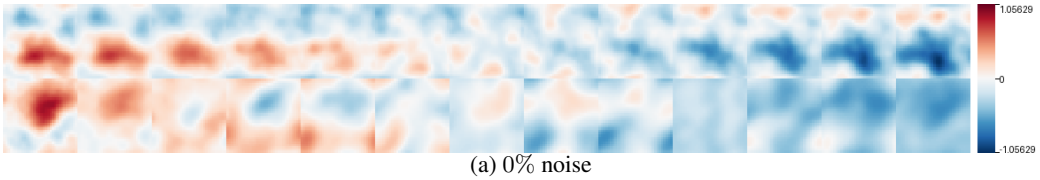
Family	**Terms**	**Count**
Constant	1	1
Linear values	u_j	C
First spatial derivatives	u_{jx}, u_{jy}	$2C$
Second spatial derivatives	u_{jxx}, u_{jyy}	$2C$
Temporal derivatives	u_{jt}, u_{jtt}	$2C$
Symmetric quadratic values	$u_j u_k,\ j \le k$	$C(C+1)/2$
Value–gradient products	$u_j u_{kx}, u_j u_{ky}$	$2C^2$

A.3 Observational Noise

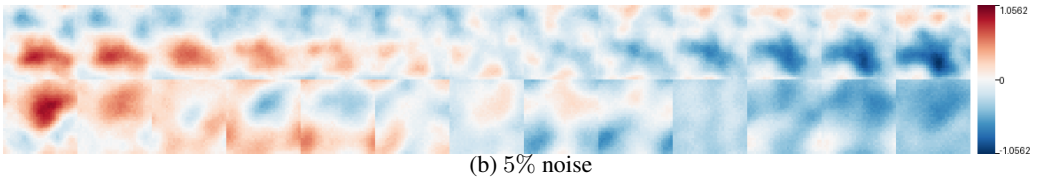
This subsection describes the Gaussian noise added to the observations when evaluating the model's robustness. For a noise level $\alpha \in \{0, 0.05, 0.10\}$, independent Gaussian noise is added to each variable

$$u_{i,\text{noisy}} = u_i + \eta_i, \eta_i(\mathbf{x}, t) \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \alpha^2 r_i^2\right) , \quad \text{where} \quad r_i = \sqrt{\text{mean}_{\mathbf{x},t}\, u_i(\mathbf{x}, t)^2} .$$

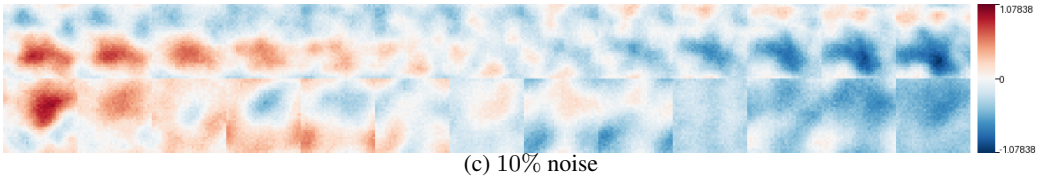
Thus, the noise standard deviation is α times the root-mean-square magnitude of the corresponding variable over the complete observations. Figure 5 visualizes how this perturbation affects the observations by comparing the same sample at 0%, 5%, and 10% noise. As the noise level increases, fine-scale fluctuations become more prominent and progressively obscure the coherent spatiotemporal structures in the clean observation.



(a) 0% noise



(b) 5% noise



(c) 10% noise

Figure 5: Effect of observational noise on representative two-variable PDE observations. Panels (a)–(c) show the same observations under 0%, 5%, and 10% Gaussian noise, respectively. Within each panel, the two rows correspond to the two variables, and the columns show the 13 time steps.

Noise is added to the raw observations before temporal chunking. O_2PDE and Qwen3-VL are trained exclusively on clean observations from the training set, without noisy-sample augmentation. For both models, the same checkpoint trained on clean observations is evaluated directly at 0%, 5%, and 10% noise, without fine-tuning or parameter updates on noisy observations.

B BASELINE CONFIGURATIONS

Unless otherwise specified, all baselines use the same validation and test sets as O_2PDE and receive the complete C-variable observations on the $32 \times 32 \times 26$ spatiotemporal grid. At each noise level, all methods use the same noisy realization of each sample. Library-based baselines are given the common candidate library in Table 7. At the same time, method-specific input representations and operations, including numerical differentiation, denoising, weak-form construction, and symbolic search, are retained and described below. The hyperparameters governing model selection, sparsity, and coefficient thresholding are selected on the validation set and frozen before test evaluation. For consistent evaluation, every recovered equation is converted to the residual form in Eq. (1). When a regression target is required, u_{ctt} is used as the response for the c-th equation and excluded from its regressor matrix; the resulting equation is then converted to residual form with the coefficient of u_{ctt} normalized to one.

PDE-FIND. We implement the PDE-FIND baseline (Rudy et al., 2017). For each system, temporal and spatial derivatives are computed independently for every variable using second-order finite differences, without additional smoothing or denoising. The candidate functions in Table 7 are then evaluated pointwise, and each target equation is recovered independently using sequentially thresholded ridge regression (STRidge). To reduce boundary-difference errors, we discard two grid points from both ends of each spatial and temporal axis. We select the STRidge tolerance from $\{0, 0.1, \ldots, 1\}$ using an 80/20 fit–validation split.

DL-PDE. We implement DL-PDE (Xu et al., 2021) as a two-stage neural differentiation and sparse-regression baseline. For each system, we train a neural network to approximate all C variable observations. The network is trained exclusively by minimizing the data mean-squared error. For every system, we randomly reserve 10% of the $32 \times 32 \times 26$ grid points for validation and use the remaining 90% as the training pool. We optimize the network using AdamW for at most 10000 steps, drawing a batch of 10000 training points with replacement at each step. Validation is performed every 10 steps and training is stopped when the validation loss fails to improve for 100 optimization steps. After training, the network is frozen. Automatic differentiation is used to compute derivatives. The candidate functions in Table 7 are then evaluated pointwise on the interior grid obtained by discarding two points from each end of every coordinate axis. For the c-th equation, u_{ctt} is excluded from the regressor matrix and used as the response. The remaining active terms and their coefficients are recovered using the same STRidge implementation and configuration as PDE-FIND.

WSINDy. We adopt WSINDy (Messenger & Bortz, 2021), using the PyWSINDy-for-PDEs implementation with a custom adapter for our data format. Rather than directly differentiating the observed fields, WSINDy transfers derivatives to compactly supported test functions and constructs a weak-form linear system through separable convolution. Each equation of each system is then solved independently by modified sequential threshold least squares (MSTLS), whose two-sided thresholding iterates until the support becomes unchanged or empty. We leave the upstream algorithm unchanged but add the project-specific candidate function mapping and use the canonical flux form so that weak-form flux terms are expanded by the product rule into the column order of Table 7.

SGA-PDE. We implement a symbolic genetic baseline following SGA-PDE (Chen et al., 2022). For each system and target equation, an individual in the population is a forest of expression trees whose leaves comprise the observed variables and their derivatives and whose internal nodes apply arithmetic, squaring, and differential operators. Each individual is evaluated on finite-difference fields, fitted with STRidge to estimate its nonzero coefficients, and evolved according to an Akaike information criterion (AIC) fitness that balances residual error and expression sparsity. The individual with the best AIC over the complete evolutionary history is selected for each equation. For evaluation against the ground truth, we expand the predicted symbolic trees into polynomials, collect algebraically identical terms, convert them to the common residual sign and coefficient scale, and project them onto the candidate function library. Terms that cannot be mapped to the predefined library are retained as out-of-library terms.

ABL-PDE. We implement the ABL-PDE baseline from the official implementation (Gao et al., 2025). For each system, ABL-PDE trains a coordinate network from scratch, computes candidate quantities by automatic differentiation, and uses a first-order-logic knowledge base in Prolog to generate admissible candidate structures. For each target equation, ZOOpt first selects a sparse subset of candidate functions. An exhaustive subspace search then identifies the smallest subset

whose residual is within a factor of that obtained with the complete selected support, after which validation-block support recovery is used to add potentially missed terms. The coefficients on the resulting support are estimated by least squares through a pseudoinverse and written back as trainable variables. We run five abductive-learning loops; the discovered PDE residual is incorporated into network training from the third loop onward. Finally, we restore the loop with the best model-selection score, which combines the field error, derivative error, and holdout PDE residual.

LLM-SFT. As an end-to-end baseline, we apply supervised LoRA fine-tuning to Qwen3-VL-2B-Instruct (Bai et al., 2025) using the official Qwen fine-tuning framework. Each PDE system is rendered as a grid of heat maps, with a shared symmetric color scale determined by the maximum absolute value over all displayed fields in that sample. The candidate function library, output JSON schema, and one example are included inline in the prompt, and the model is trained to generate the JSON file containing only nonzero terms and their coefficients. LoRA updates only the $q/k/v/o$ projections in the language model, while the vision tower and multimodal projector remain frozen. At inference time, each system is processed by one greedy generation without explicit differentiation, numerical candidate evaluation, sparse regression, retries, or voting. The generated text is parsed and mapped to the candidate function library, while invalid JSON or predictions with an incorrect matrix shape are scored as all-zero coefficient matrices.

C Evaluation Metrics

This section defines the metrics used to evaluate two complementary aspects of the inverse PDE problem: structure identification and coefficient identification. We first introduce the common notation, then define the metrics in Table 8, and finally present the term-level metrics used in Figure 2.

Table 8: Evaluation metrics.

Aspect	Metrics	Description
Structure Identification	F1 ↑	Harmonic mean of precision and recall computed with micro-aggregation. It evaluates term-level structure identification by balancing missed and spurious terms.
	EM (%) ↑	Percentage of equations whose structure is exactly recovered. It evaluates equation-level structure identification by requiring all terms in an equation to be correctly identified.
Coefficient Identification	L2RE ↓	Relative ℓ_2 coefficient error over the union of nonzero ground-truth and predicted terms. It evaluates coefficient identification accuracy.

C.1 Common Notation

Let $\mathcal{E}$ be the set of evaluated equations. For each equation $e \in \mathcal{E}$, let $\mathcal{T}_e^*$ and $\hat{\mathcal{T}}_e$ denote the sets of nonzero canonical terms in the ground-truth and predicted equations, respectively. For an equation whose predefined candidate function library contains N_e terms, let $\mathcal{L}_e = \{1, \dots, N_e\}$ be the corresponding index set. For each candidate function ϕ_j, let $c_{e,j}^*$ and $\hat{c}_{e,j}$ denote its ground-truth and predicted coefficients, with the coefficient set to zero when the corresponding term is absent. The associated support indicators are

$$y_{e,j} = \mathbb{I}\left(c_{e,j}^* \neq 0\right) , \quad \text{and} \quad \hat{y}_{e,j} = \mathbb{I}\left(\hat{c}_{e,j} \neq 0\right) .$$

C.2 Metrics

The aggregate metrics characterize reconstruction performance over the complete evaluation set. F1 with micro aggregation and Exact Match (EM) evaluate term-level and equation-level structure identification, respectively, whereas L2RE evaluates coefficient identification.

F1 with micro aggregation. To compute the F1 in Table 8, we first pool the confusion counts over all equations and their candidate function libraries:

$$\mathrm{TP} = \sum_{e\in\mathcal{E}} \sum_{j\in\mathcal{L}_e} y_{e,j}\hat{y}_{e,j} \ , \qquad \mathrm{FP} = \sum_{e\in\mathcal{E}} \sum_{j\in\mathcal{L}_e} (1-y_{e,j})\hat{y}_{e,j} \ ,$$
$$\text{and} \quad \mathrm{FN} = \sum_{e\in\mathcal{E}} \sum_{j\in\mathcal{L}_e} y_{e,j}\,(1-\hat{y}_{e,j}) \ .$$

The resulting score is

$$\mathrm{F1} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \ .$$

Exact Match. EM evaluates whether the complete structure of each equation is recovered

$$\mathrm{EM} = \frac{1}{|\mathcal{E}|} \sum_{e\in\mathcal{E}} \mathbb{I}\left(\hat{\mathcal{T}}_e = \mathcal{T}_e^*\right) \ .$$

Thus, any missed term or additional term, including a term outside the predefined candidate function library, makes the equation a mismatch. We report $100 \times \mathrm{EM}$ as a percentage in all tables.

Relative ℓ_2 coefficient error. L2RE evaluates coefficient identification accuracy. For each equation, define the union of its nonzero ground-truth and predicted canonical terms as $\mathcal{S}_e = \mathcal{T}_e^* \cup \hat{\mathcal{T}}_e$. For any canonical term ϕ, let $c^*_{e,\phi}$ and $\hat{c}_{e,\phi}$ be its ground-truth and predicted coefficients, respectively, using zero for an absent term. The coefficient vectors aligned over $\mathcal{S}_e$ are

$$\mathbf{c}^*_{e,\mathcal{S}_e} = \left(c^*_{e,\phi}\right)_{\phi\in\mathcal{S}_e} \ , \quad \text{and} \quad \hat{\mathbf{c}}_{e,\mathcal{S}_e} = \left(\hat{c}_{e,\phi}\right)_{\phi\in\mathcal{S}_e} \ .$$

The L2RE is computed for each equation over this union and then averaged across equations. This definition penalizes missed terms through nonzero entries in $\mathbf{c}^*_{e,\mathcal{S}_e}$ and additional predicted terms through nonzero entries in $\hat{\mathbf{c}}_{e,\mathcal{S}_e}$

$$\mathrm{L2RE} = \frac{1}{|\mathcal{E}|} \sum_{e\in\mathcal{E}} \frac{\left\|\hat{\mathbf{c}}_{e,\mathcal{S}_e} - \mathbf{c}^*_{e,\mathcal{S}_e}\right\|_2}{\left\|\mathbf{c}^*_{e,\mathcal{S}_e}\right\|_2} \ .$$

The aggregate metrics can mask term-specific behavior. Thus, we compute the term-level metrics in Figure 2 independently for each candidate function rather than pooling over the complete library.

Term-level F1. For a fixed term ϕ_j, let $\mathcal{E}_j = \{e \in \mathcal{E} : j \in \mathcal{L}_e\}$ be the equations in the evaluated subset whose candidate function library contains ϕ_j. Its confusion counts and term-level F1 are

$$\mathrm{TP}_j = \sum_{e\in\mathcal{E}_j} y_{e,j}\hat{y}_{e,j} \ , \qquad \mathrm{FP}_j = \sum_{e\in\mathcal{E}_j} (1-y_{e,j})\hat{y}_{e,j} \ ,$$
$$\mathrm{FN}_j = \sum_{e\in\mathcal{E}_j} y_{e,j}\,(1-\hat{y}_{e,j}) \ , \quad \text{and} \quad \mathrm{F1}_j = \frac{2\mathrm{TP}_j}{2\mathrm{TP}_j+\mathrm{FP}_j+\mathrm{FN}_j} \ .$$

Thus, $\mathrm{F1}_j$ pools only the occurrences and predictions of term ϕ_j across equations, and it is not an average of per-equation scores.

Term-level L2RE. For term ϕ_j, define the union of the equations in which it has a nonzero ground-truth or predicted coefficient as

$$\mathcal{U}_j = \left\{e \in \mathcal{E}_j : c^*_{e,j} \neq 0\right\} \cup \left\{e \in \mathcal{E}_j : \hat{c}_{e,j} \neq 0\right\} \ .$$

The corresponding term-level L2RE is the relative ℓ_2 coefficient error computed over this union

$$\mathrm{L2RE}_j = \frac{\left(\sum_{e\in\mathcal{U}_j} \left(\hat{c}_{e,j} - c^*_{e,j}\right)^2\right)^{1/2}}{\left(\sum_{e\in\mathcal{U}_j} \left(c^*_{e,j}\right)^2\right)^{1/2}} \ .$$

Therefore, $\mathrm{L2RE}_j$ penalizes both missed occurrences and spurious predictions of term ϕ_j. It is reported only when term ϕ_j occurs at least once in the evaluated ground-truth equations, so that the denominator is nonzero.

C.3 Metric Availability Across Baselines

F1 requires predictions to be scored over a candidate function library and is reported only for methods whose output space is that library. SGA-PDE is instead evaluated in its native open-form setting, in which it may generate out-of-library terms. For SGA-PDE, we report EM, which requires the complete structure to be exactly recovered and treats any generated out-of-library term as a mismatch, and L2RE, which is computed over the union of nonzero ground-truth and predicted terms.

D Implementation Details

This section presents the training details of O_2PDE and the shared configurations of the models implemented in our experiments. Table 9 collects the numerical architecture, optimization, and loss settings. All models are optimized with AdamW and cosine learning-rate decay. Training uses bfloat16 mixed precision with FP32 master weights. The largest model contains 101.81M parameters and is trained on eight NVIDIA GeForce RTX 5090 GPUs.

Table 9: Shared numerical settings of the O_2PDE model configurations.

Setting	**Value**
Latent dimension (d)	256
Attention heads	8
FFN dimension	1024
Features per variable (N_x)	96
Query hidden dimension	256
Temporal chunks (K)	4
Overlap ratio	0.5
Learning rate	10^{-4}
Weight decay	10^{-4}
Final learning rate	10^{-6}
ASL exponents (γ_-/γ_+)	2/1
ASL clipping	0.05
Huber transition	0.2
Classification weight (λ_{cls})	1
Regression weight (λ_{reg})	0.5
Consistency weight (λ_{con})	0.1
Coefficient consistency (γ)	0.1

E Detailed Results

Figure 6 examines how the magnitude of an active ground-truth coefficient affects structure identification across candidate-function families. Each cell reports the F1 score for the corresponding coefficient-magnitude interval, while gray cells denote intervals containing no ground-truth instances. For most function families, terms with coefficients below 0.03 are difficult to identify, whereas the F1 score improves substantially once the magnitude exceeds 0.1. Second order spatial derivatives are a notable exception, remaining reliably identifiable even at small magnitudes. These results empirically support the conjecture in Section 5 that candidate functions with small coefficients exert weaker effects on the observed dynamics and are therefore more difficult to distinguish from inactive candidate functions.

Figures 7 and 8 provide complementary bar-chart views of the term-level performance summarized by the radar plots in Figure 2. Within each row, candidate functions are grouped by function family. Figure 7 reports precision, recall, and F1 using three bars for each term, whereas Figure 8 reports the corresponding L2RE on clean observations. Together, these visualizations make the term-wise performances directly visible. O_2PDE consistently achieves high structure identification and coefficient identification accuracy for derivative-related functions. In contrast, derivative-free functions remain challenging. These term-wise results provide additional detail for Subsection 4.2.

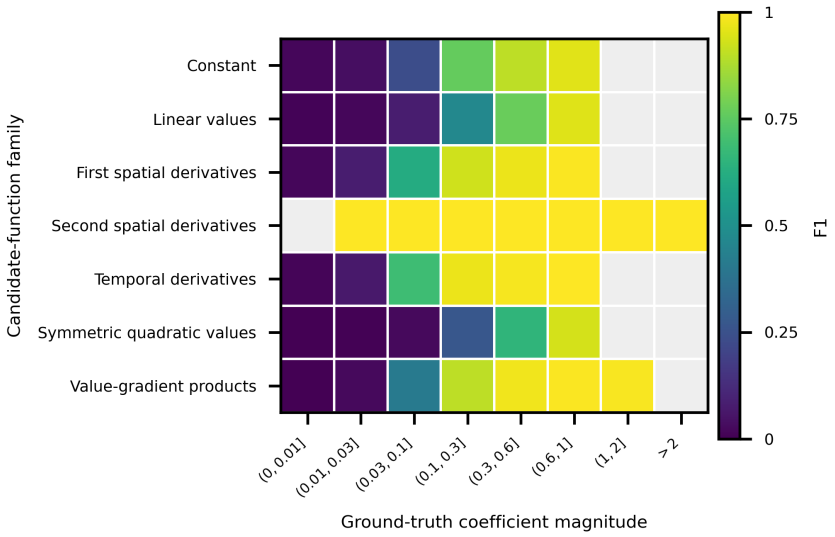


Figure 6: Performance of identification with respect to the magnitude of the coefficients.

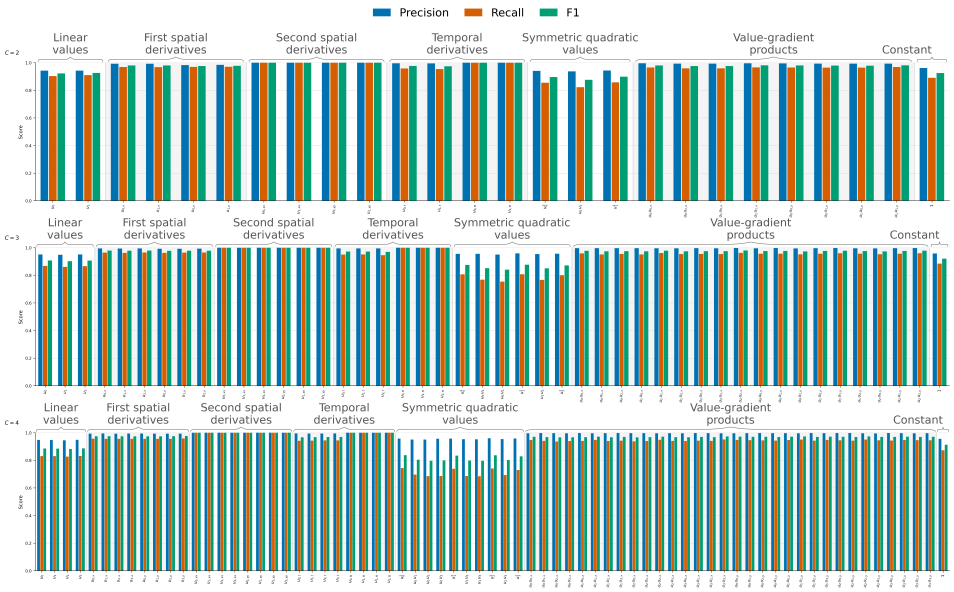


Figure 7: Bar-chart visualization of the term-level precision, recall, and F1.

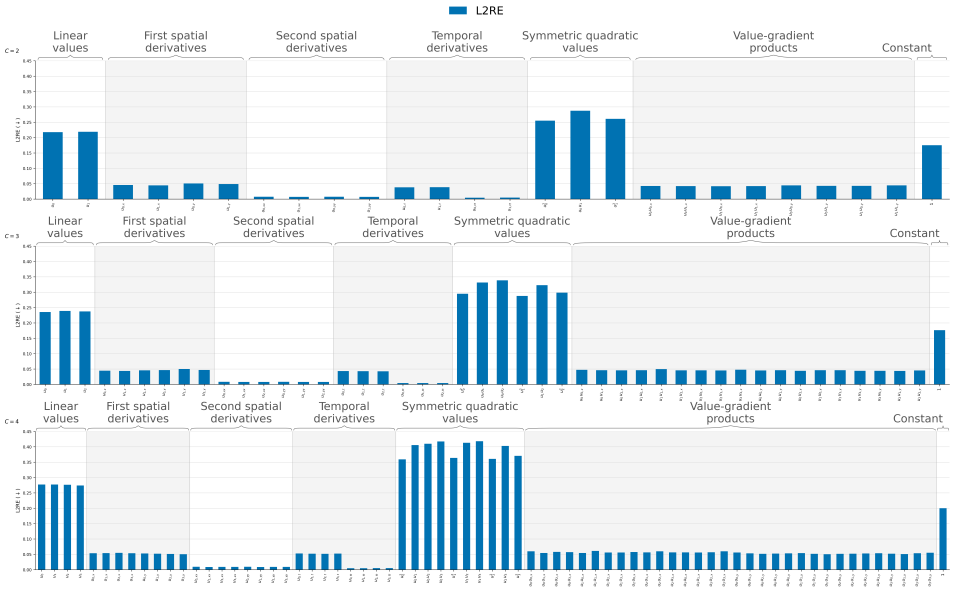


Figure 8: Bar-chart visualization of the term-level L2RE.