PROSE: Predicting Operators and Symbolic Expressions using Multimodal Transformers[*]

Yuxuan Liu[†] Zecheng Zhang[‡] Hayden Schaeffer[*]

Abstract

Approximating nonlinear differential equations using a neural network provides a robust and efficient tool for various scientific computing tasks, including real-time predictions, inverse problems, optimal controls, and surrogate modeling. Previous works have focused on embedding dynamical systems into networks through two approaches: learning a single solution operator (i.e., the mapping from input parametrized functions to solutions) or learning the governing system of equations (i.e., the constitutive model relative to the state variables). Both of these approaches yield different representations for the same underlying data or function. Additionally, observing that families of differential equations often share key characteristics, we seek one network representation across a wide range of equations. Our method, called **Pr**edicting **O**perators and **S**ymbolic **E**xpressions (PROSE), learns maps from multimodal inputs to multimodal outputs, capable of generating both numerical predictions and mathematical equations. By using a transformer structure and a feature fusion approach, our network can simultaneously embed sets of solution operators for various parametric differential equations using a single trained network. Detailed experiments demonstrate that the network benefits from its multimodal nature, resulting in improved prediction accuracy and better generalization. The network is shown to be able to handle noise in the data and errors in the symbolic representation, including noisy numerical values, model misspecification, and erroneous addition or deletion of terms. PROSE provides a new neural network framework for differential equations which allows for more flexibility and generality in learning operators and governing equations from data.

1 Introduction

Differential equations are important tools for understanding and studying nonlinear physical phenomena and time-series dynamics. They are necessary for a multitude of modern scientific and engineering applications, including stability analysis, state variable prediction, structural optimization, and design. Consider parametric ordinary differential equations (ODEs), i.e. differential equations whose initial conditions and coefficients are parameterized by functions with inputs from some distribution. We can denote the system by $\frac{d\boldsymbol{u}}{dt} = f\left(\boldsymbol{u}; a_s(t)\right)$, where $\boldsymbol{u}(t) \in \mathbb{R}^d$ are states, and $a_s(t)$ is the parametric function with input parameter s. For example, $a_s(t)$ could be an additive forcing term where s follows a normal distribution. The goal of computational methods for parametric ODEs is to evaluate the solution given a new parametric function, often with the need to generalize to larger parameter distributions, i.e. out-of-distribution predictions.

Recently, *operator learning* has been used to encode the operator that maps input functions $a_s(-)$ to the solution $\boldsymbol{u}(-; a_s(-))$ through a deep network, whose evaluation is more cost-efficient

[*]The paper is also avariable at: `https://arxiv.org/abs/2309.16816`. The code is avariable at: `https://github.com/felix-lyx/prose`.

[†]Department of Mathematics, UCLA, Los Angeles, CA 90095.

[‡]Department of Mathematics, Florida State University, Tallahassee, FL 32304.

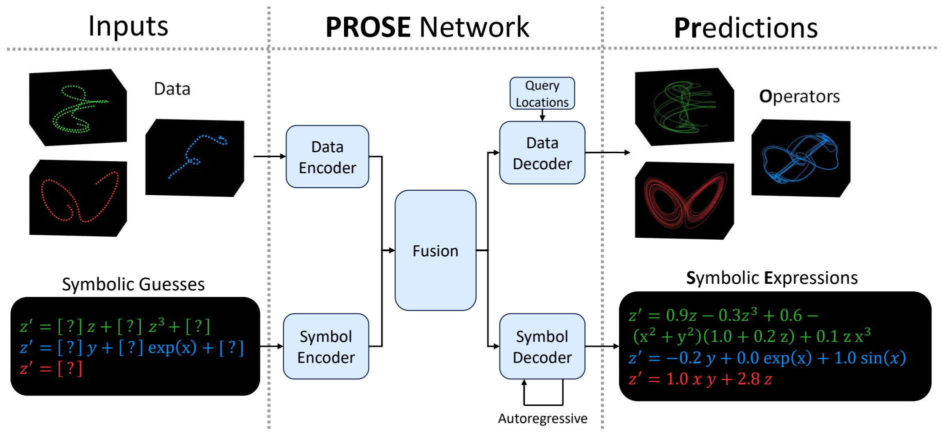

Inputs
PROSE Network
Predictions
Data
Data Encoder
Query Locations
Data Decoder
Operators
Fusion
Symbolic Guesses
$z' = [?]\, z + [?]\, z^3 + [?]$
$z' = [?]\, y + [?]\, \exp(x) + [?]$
$z' = [?]$
Symbol Encoder
Symbol Decoder
Autoregressive
Symbolic Expressions
$z' = 0.9z - 0.3z^3 + 0.6 - (x^2 + y^2)(1.0 + 0.2\, z) + 0.1\, z\, x^3$
$z' = -0.2\, y + 0.0\, \exp(x) + 1.0\, \sin(x)$
$z' = 1.0\, x\, y + 2.8\, z$

Figure 1: **PROSE network illustration.** The inputs and outputs (predictions) are multi-modal, each including numerical values (data) and symbolic expressions (governing equations). Here we include just the third term in the governing equations for simpler visualization.

than fully simulating the differential equations [11, 34, 36, 41, 69]. An advantage of operator learning compared to conventional networks is that the resulting approximation captures the mapping between functions, rather than being limited to fixed-size vectors. This flexibility enables a broader range of downstream tasks to be undertaken, especially in multi-query settings. However, operator learning is limited to training solutions for an individual differential equation. In particular, current operator learning methods do not benefit from observations of similar systems and, once trained, do not generalize to new differential equations.

Problem Statement We consider the problem of encoding multiple ODEs and parametric functions, for use in generalized prediction and model discovery. Specifically, we are given N ODEs f_j, and parametric functions $a_s^j(t)$, with the goal of constructing a single neural network to both identify the system and the operator from parametric functions $a_s^j(-)$ to solutions. Consider a family of differential equations indexed by $j = 1, \cdots, N$, with the form $\frac{d\boldsymbol{u}}{dt} = f_j\left(\boldsymbol{u}; a_s^j(t)\right)$, where the solutions are denoted by $\boldsymbol{u}_j(-; a_s^j(-))$. The solution operator G^j encodes the solution's dependence on a_s^j and corresponds to the j^{th} ODE. When utilizing standard operator learning, it becomes necessary to train separate deep networks for each of the N equations. That approach can quickly become impractical and inefficient, especially in the context of most nonlinear scientific problems.

This work introduces a multimodal framework for simultaneously encoding multiple operators for use in predicting states at query locations and discovering the governing model that represents the equations of motion describing the data. For data prediction, a novel transformer-based approach which we call *multi-operator learning* is employed. This entails training the network to learn the solution operator across a set of distinct parametric dynamical systems. In other words, the network learns a single operator $\bar{G}$ that represents the family of mappings $\{G^1, \cdots, G^N\}$ by leveraging shared characteristics among their features. This should also allow the network to predict new operators that share commonalities with those from the family of operators used in training, i.e. generalize to new operators. During testing or prediction, the governing equations (i.e. the mathematical equations defining the dynamics of dependent variables for a given data sequence) are not known, so the algorithm also produces a symbolic expression using a generative model. In other words, the network learns a syntax for representing and articulating differential equations. In this way, the approach yields a network capable of evaluating dependent variables at query locations over wide parameter sets and also "writes" the mathematical differential equation associated to the data. This can be viewed as a large language model for differential equations.

Main Contributions The Predicting Operators and Symbolic Expression (PROSE) framework introduces a new approach to learning differential equations from data. The key components of the architecture are illustrated are Figure 1. The main contributions and novelty are summarized below.

- PROSE is the first method to generate both the governing system and an operator network from multiple distinct ODEs. It is one of the first multi-operator learning approaches.
- PROSE incorporates a new modality through a fusion structure. Unlike text modality or labels, the symbolic expression can accurately generate the system solution.
- The network architecture introduces new structural elements, including a fusion transformer that connects the data and embedded symbols.
- We demonstrate accuracy in generating valid ODEs (validity is of $> 99.9\%$ on in-distribution tests and $> 97.89\%$ on out-of-distribution predictions), showing that PROSE can generate new ODEs from data.

2 Related Works

PROSE is both a multi-operator learning and a model discovery approach. We summarize these two distinct research areas in this section.

Operator Learning Operator learning [10, 11, 34, 36, 41, 69] studies neural network approximations to an operator $G : U \to V$, where U and V are function spaces. This approach holds significant relevance in various mathematical problems, including the solution of parametric PDEs [5, 29], control of dynamical systems [37, 68], and multi-fidelity modeling [1, 43, 70]. Operator learning has gained substantial popularity within the mathematical and scientific machine learning community, with applications in engineering domains [46]. Currently, methods for neural operators focus on constructing a single operator, e.g. learning the map from the initial conditions or parameters of a physical system to the solution at a terminal time.

In [10, 11], the authors extended the universal approximation theory from function approximation [3, 13, 24] to operators. This work paved the way for the modern development of deep neural operator learning (DON) as seen in [36, 41, 42]. Building upon the principles of [11], [69] further expanded this approach by constructing operator networks that remain invariant to the input/output function discretizations. The noisy operator learning and optimization is studied in [36]. Another operator approach is the Fourier neural operators (FNO) [34, 62], which use Fourier transformations and their inverses in approximating operators through kernel integral approximations. Comparative analysis can be found in [42, 69].

The multi-input-output network (MioNet) [23] extends operator learning to handle multiple input/output parametric functions within the single operator framework. Recently, the In-Context Operator Network (ICON) [66] was developed for multi-operator learning using data and equation labels (one-hot encoding) as prompts and a test label during inference. This was later extended to include multimodal inputs by allowing captions which are embedded into the input sequence using a pre-trained language model [67]. Multi-operator learning has significant challenges, especially when encoding the operators or when addressing out-of-distribution problems (i.e. those that extend beyond the training dataset).

Learning Governing Equations Learning mathematical models from observations of dynamical systems is an essential scientific task, resulting in the ability to analyze relations between variables and obtain a deeper understanding of the laws of nature. In the works [6, 55], the authors introduced a symbolic regression approach for learning constitutive equations and other physically relevant equations from time-series data. The SINDy algorithm, introduced in [7], utilizes a dictionary of candidate features that often includes polynomials and trigonometric functions. They developed an iterative thresholding method to obtain a sparse model, with the goal of achieving a parsimonious representation of the relationships among potential model terms. SINDy has found applications in a wide range of problems and formulations, as demonstrated in [8, 21, 25, 44, 49, 56]. Sparse optimization techniques for learning partial differential equations were developed in [50] for spatio-temporal data. This approach incorporates differential operators into the dictionary, and the governing equation is trained using the LASSO method. The ℓ^1-based approaches offer statistical guarantees with respect to the error bounds and equation recovery rates. These methods have been further refined and extended in subsequent works, including [39, 51–54]. In [12], the Physics-Informed Neural Network with Sparse Regression (PINN-SR) method for discovering PDE models demonstrated that the equation learning paradigm can be leveraged within the PINNs [27, 31, 48] framework to train models from scarce data. The operator inference technique [47] approximates high-dimensional differential equations by first reducing the data-dimension to a small set of variables and training a lower-dimensional ODE model using a least-squares fit over polynomial features. This is particularly advantageous when dealing with high-dimensional data and when the original differential equations are inaccessible.

3 Methodology

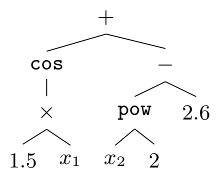
The main ingredients of PROSE include symbol embedding, transformers, and multimodal inputs and outputs. We summarize these key elements in this section.

Transformers A transformer is an attention-driven mechanism that excels at capturing longer-term dependencies in data [4, 14, 61]. The vanilla transformer uses a self-attention architecture [2, 64], enabling it to capture intricate relationships within lengthy time series data. Specifically, let us denote the input time series data as $X \in \mathbb{R}^{n \times d}$, where n is the number of time steps and d is the dimension of each element in the time series. Self-attention first computes the projections: query $Q = XW^Q$, key $K = XW^K$ and value $V = XW^V$, where $W^Q \in \mathbb{R}^{d \times d_k}$, $W^K \in \mathbb{R}^{d \times d_k}$, and $W^V \in \mathbb{R}^{d \times d_v}$. It then outputs the context $C \in \mathbb{R}^{n \times d_v}$ via $C = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$, where the softmax function is calculated over all entries of each row. Self-attention discovers relationships among various elements within a time sequence. Predictions often depend on multiple data sources, making it crucial to understand the interactions and encode various time series data (see Section 3 for details). This self-attention idea has driven the development of the cross-attention mechanism [33, 40, 60]. Given two input time series data X, Y, cross-attention computes the query, key, and value as $Q = XW^Q$, $K = YW^K$, and $V = YW^V$. In the case where Y represents the output of a decoder and X represents the output of an encoder, the cross-attention, which directs its focus from X to Y, is commonly referred to as encoder-decoder attention [61]. Encoder-decoder attention serves as a crucial component within autoregressive models [20, 33, 61]. The autoregressive model operates by making predictions for a time series iteratively, one step at a time. To achieve this, it utilizes the previous step's generated output as additional input for the subsequent prediction. This approach has demonstrated the capacity for mitigating accumulation errors [19], which makes it desirable for longer-time predictions.

Multimodal Machine Learning Multimodal machine learning (MML) trains models using data from heterogeneous sources [33, 40, 57, 59, 65]. Of major interest in this topic are methods for the fusion of data from multiple modalities, the exploration of their interplay, and the development of corresponding models and algorithms. For instance, consider the field of visual-language reasoning [32, 57, 59], where the utilization of visual content, such as images or videos, with the semantics of language [59] associated with these visual elements, such as captions or descriptions, leads to the development of models with richer information [32]. Another illustrative example is that of AI robots, which use multimodal sensors, including cameras, radar systems, and ultrasounds, to perceive their environment and make decisions [18, 38]. In mathematical applications, researchers employ multiscale mathematical models [17], where each modality is essentially characterized by varying levels of accuracy, to train a single model capable of predicting multiscale differential equations effectively.

Operator Learning Structure The authors in [11] established a universal approximation theory for continuous operators, denoted by G. Particularly, they showed that the neural operator $G_\theta(u)(t) = \sum_{k=1}^{K} b_k(t)p_k(\hat{u})$ can approximate $G(u)(t)$ for t in the output function domain (under certain conditions). Here $p(\cdot)$ and $b(\cdot)$ are neural networks which are called the branch and trunk [41], and $\hat{u}$ is a discretized approximation to the input function u. In our applications, these input functions u correspond to ODE solutions sampled in the input intervals, and the output functions are solutions over larger intervals. Based on the output-discretization invariance property of the network [42, 69], the output of the operator network can be queried at arbitrary timepoints, allowing predictions of the solution at any location.

Equation Encoding via Polish Notation Mathematical expressions can be encoded as trees with operations and functions as nodes, and constants and variables as leaves [22, 35]. For instance, the tree on the right represents the expression $\cos(1.5x_1) + x_2^2 - 2.6$.



Trees provide natural evaluation orders, eliminating the need to use parentheses or spaces. Under some additional restrictions (e.g. $1 + 2 + 3$ should be processed as $1 + (2 + 3)$, $-1 \times x$ is equivalent to $-x$), there is a one-to-one correspondence between trees and mathematical expressions. For these reasons, trees provide an unambiguous way of encoding equations. While there are existing tree2tree methods [16, 58], they are usually slower than seq2seq methods at training and inference time. The preorder traversal is a consistent way of mapping trees to sequences, and the resulting sequences are known as Polish or Prefix notation, which is used in our equation encoder. For the above expression $\cos(1.5x_1) + x_2^2 - 2.6$, its Polish notation is given by the sequence `[+ cos` $\times$ `1.5` x_1 `− pow` x_2 `2 2.6]`. Operations such as `cos` are treated as single words and are not further tokenized, but they are trainable. In comparison to LaTeX representations of mathematical expressions, Polish notations have shorter lengths, simpler syntax, and are often more consistent. Note that in [22, 35], binary trees of depth-3 are used to generate symbolic approximations directly for the solution of a single differential equation.

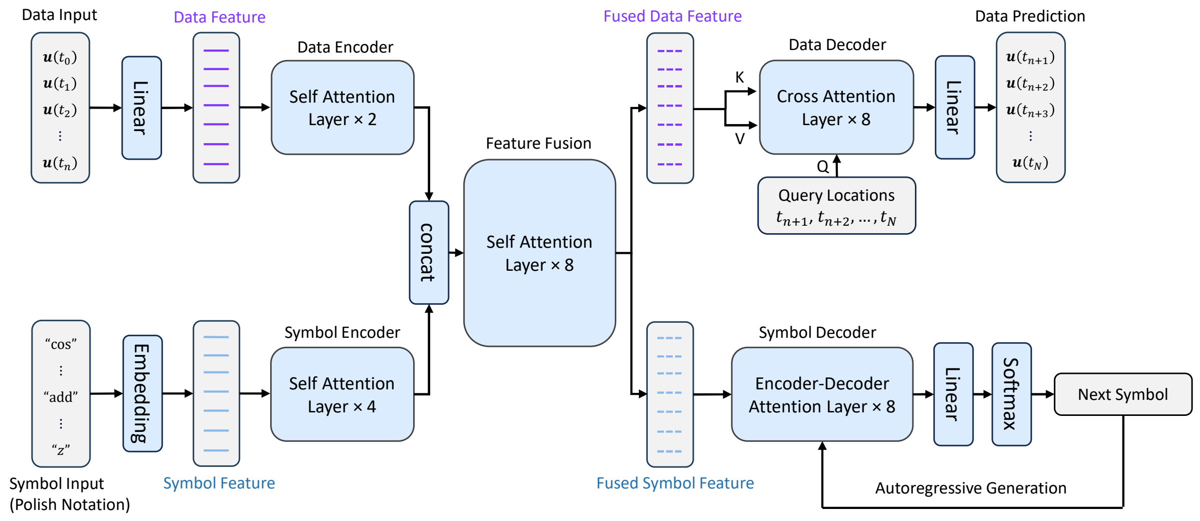


Figure 2: **PROSE architecture and the workflow.** Data Input and Symbol Input are embedded into Data Feature and Symbol Feature respectively before encoding and fusion through Feature Fusion. PROSE uses Cross-Attention to construct the operator (upper-right structure) from Fused Data Feature, and evaluate it at Query Locations. PROSE generates symbolic expressions in the lower-right portion autoregressively. Attention blocks are displayed in Appendix C.

Following [9, 15, 26, 30], to have a reasonable vocabulary size, floating point numbers are represented in their base-10 notations, each consisting of three components: sign, mantissa, and exponent, which are treated as words with trainable embedding. For example, if the length of the mantissa is chosen to be 3, then $2.6 = +1 \cdot 260 \cdot 10^{-2}$ is represented as `[+ 260 E-2]`. For vector-valued functions, a dimension-separation token is used, i.e. $\boldsymbol{f} = (f_1, f_2)$ is represented as "$f_1 \mid f_2$". Similar to [9, 15, 26, 30], our vocabulary is also of order 10^4 words.

3.1 Model Overview

Our network uses hierarchical attention for feature processing and fusion, and two transformer decoders for two downstream tasks. Figure 2 provides an overview of the architecture. The PROSE architecture contains five main components trained end-to-end: data encoder, symbol encoder, feature fusion, data decoder, and symbol decoder.

Encoders Two separate transformer encoders are used to obtain domain-specific features. Given numerical data inputs and symbolic equation guesses (possibly empty or erroneous), the data encoder and symbol encoder first separately perform feature aggregation using self-attention. For a data input sequence $\boldsymbol{u}(t_0), \cdots, \boldsymbol{u}(t_n)$, each element $\boldsymbol{u}(t_i)$, together with its time variable t_i, goes through a linear layer to form the Data Feature (purple feature sequence in Figure 2). PROSE then uses self-attention to further process the Data Feature, where the time variables t_i serve as the positional encoding.

The symbolic input (in Polish notation) is a standard word sequence, which can be directly processed with self-attention layers. The word embedding (for operations, sign, mantissa, etc.) is randomly initialized and trainable. Sinusoidal positional encoding [61] is used for the symbol encoder.

Feature Fusion Hierarchical attention (multi-stream to one-stream) is used in this model for feature fusion. Separately-processed data and symbol features are concatenated into a feature sequence, and further processed through self-attention layers where modality interaction occurs. Following [28], a learnable modality-type embedding is added to the fused features, explicitly signaling to the model which parts of the sequence are from which modality. Positional encoding is not needed since it is already included in the individual encoders.

Data Decoder The data decoder constructs the operator via the cross-attention mechanism, establishing a link between the input-encoded time sequence (fused data features) and the output functions. The query locations, representing the independent variables of these output functions, serve as the evaluation points. Importantly, these query locations operate independently of each other, meaning that assessing the operator at one point, t_i, does not impact the evaluation of the operator at another point, t_j. As a result, the time and space complexity scales linearly with the number of query locations. In addition, since the evaluation points are independent of the network generation, this resembles the philosophy of the branch and trunk nets, see Operator Learning Structure in Section 3.

Symbol Decoder The symbol decoder is a standard encoder-decoder transformer, where the fused symbol feature is the context for generation. The output equation is produced using an autoregressive approach [19, 61]: it starts with the start-of-sentence token and proceeds iteratively, generating each term of the equation based on prior predictions, until it encounters the end-of-sentence token for that specific equation. During evaluation time, greedy search (iterative selection of symbol with maximum probability) is used for efficient symbol generation. While beam search [63] can be used to improve the performance (e.g. percentage of valid expression outputs), we empirically find that greedy search is sufficient for obtaining valid mathematical expressions using the Polish notation formulation.

4 Experiments

We detail the numerical experiments and studies in this section. We created a dataset of 15 distinct multi-dimensional nonlinear ODEs. To verify the performance of the PROSE approach, we conduct four case studies (Table 2) with different symbolic and data inputs (Table 1). Additionally, in the ablation study, we confirm that the inclusion of symbolic equation information enhances the accuracy of the data prediction. Hyperparameters and experimental conditions can be found in Appendix A.

Dataset The dataset is created from a dictionary of 15 distinct ODEs with varying dimensions: twelve 3D systems, two 4D systems, and one 5D system. To generate samples, we uniformly sample the coefficients of each term in the ODEs from the range $[F - 0.1F, F + 0.1F]$, where F represents the value of interest. We refer to Appendix B for the details.

The goal is to accurately predict the solutions of ODEs at future timepoints only using observations of a few points along one trajectory. We do not assume knowledge of the governing equation and thus the equations are also trained using the PROSE approach. The operator's input function is the values along the trajectories, discretized using a 64-point uniform mesh in the interval $[0, 2]$. The target operator maps this input function to the ODE solution in the interval $[2, 6]$. To assess PROSE's performance under noisy conditions, we introduce 2% Gaussian noise directly to the data samples.

Table 1: **Experiment settings.** Data-noise: additive noise on data. Unknown coefficients: replace the input equation coefficients with placeholders. Term deletion: omit a term in the target equation with 15% chance. Term addition: add an erroneous term with 15% chance. For the last test, all data inputs are padded to the maximum equation dimension. "Unknown expressions" means that the coefficients are unknown and there are terms added and removed.

Experiments (Expression Type)	Data-Noise	Unknown Coefficients	Term Deletion	Term Addition	# ODEs
Known	✓	✗	✗	✗	12
Skeleton	✓	✓	✗	✗	12
Unknown (3D)	✓	✓	✓	✓	12
Unknown (Multi-D)	✓	✓	✓	✓	15

The training dataset contains 512K examples, where 20 initial conditions are sampled to generate solution curves for each randomly generated system. The validation dataset contains 25.6K examples, where 4 initial conditions are sampled for each ODE system. The testing dataset contains 102,400 examples, where 4 initial conditions are sampled for each ODE system. The training dataset and the testing dataset contain the same number of ODE systems. In terms of practical applications, given test cases with unknown models, we are free to continue to augment the training and validation sets with any ODE, thus the dataset can be made arbitrarily large.

To test the performance of the equation prediction, we corrupt the input equation by randomly replacing, deleting, and adding terms. The terminologies and settings are found in Table 1.

Evaluation Metrics As PROSE predicts the operator and learns the equation, we present three metrics to evaluate the model performance for solution and equation learning. For data prediction, the relative L^2 error is reported. For the expression outputs (symbolic sequences in Polish notation), a decoding algorithm is used to transform the sequences into trees representing functions. The percentage of outputs that can be transformed into valid mathematical expressions is reported. Valid expressions (which approximate the velocity maps of ODE systems) are evaluated at 50 points in $\mathbb{R}^d$ where each coordinate is uniformly sampled in $[-5, 5]$ (i.e. a Monte Carlo estimate) and the relative L^2 error is reported. Here d is the dimension of the ODE system. More specifically, suppose $f(\mathbf{u})$ and $\hat{f}(\mathbf{u})$ are true and PROSE-generated ODE velocity maps, we report the average relative L^2 error computed at sampled points: $\frac{\|f-\hat{f}\|_2}{\|f\|_2}$.

4.1 Results

We observe in Table 2 that all experiments, even those corrupted by noise or random terms, achieve low relative prediction errors ($< 5.7\%$). The data prediction error decreases as we relax the conditions on the symbolic guesses, i.e. when the equations are "Unknown" 5.7% to "Known" 2.94%. Note in the case that the equations are "Known", we expect that the equations behave more like labels for the dataset. Moreover, the low expression error ($< 2.1\%$) shows PROSE's ability to correct and predict accurate equations, even when erroneous ODE equations are provided.

Data vs. Equation Prediction. We present the results of 10K testing samples in the "Unknown (3D)" experiment in Table 3. We see that the data prediction (whose features are influenced by the symbolic terms) is more accurate than using the learned governing equation

Table 2: **Performance of the model trained with different input expression types.** The two relative prediction errors are for interval $[2, 4]$ and $[2, 6]$, respectively.

Experiments (Expression Type)	Relative Prediction Errors (%)	Relative Expression Error (%)	Percentage of Valid Expressions (%)
Known	2.74, 2.94	0.00	100.00
Skeleton	3.39, 4.59	2.10	99.98
Unknown (3D)	3.43, 4.63	2.11	99.95
Unknown (Multi-D)	3.95, 5.66	1.88	99.94

directly. This shows the value of constructing a data prediction component rather than only relying on the learned governing equation. However, as in [26], the predicted equations can be further refined using optimization techniques, typically Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, where the predicted expression parameters can be used as a close initial guess.

Table 3: **Performance of data decoder output and symbol decoder output plus the backward differentiation formula (BDF method).**

Prediction Generation Method	Relative Prediction Error (%)	Percentage of Valid Expression Outputs (%)
Data decoder output	4.59	99.96
Symbol decoder output + BDF method	14.69	

Out-of-distribution Case Study. We study our model's ability to generalize beyond the training distribution. Specifically, we test on datasets whose parameters are sampled from a large interval $[F - \lambda F, F + \lambda F]$, where F represents a value of interest. We choose $\lambda = 0.15, 0.20$, which are greater than the training setting $\lambda = 0.10$. The results are shown in Table 4. This shows that the approach can be used for prediction even in the case where the parameter values were not observed during training time.

Table 4: **Out-of-distribution Testing Performance.** Relative prediction errors are reported for intervals $[2, 4]$ and $[2, 6]$, respectively.

Parameter Sample Relative Range λ	Relative Prediction Errors (%)	Relative Expression Error (%)	Percentage of Valid Expression Outputs (%)
0.10	3.43, 4.63	2.11	99.95
0.15	3.89, 5.71	3.21	99.44
0.20	4.94, 7.66	4.83	97.89

Ablation Study. Since the model is multimodal in both inputs and outputs, we investigate the performance gains by using the equation embedding in the construction of the features. In particular, we compare the performance of the full PROSE model with multimodal input/output (as shown in Figure 2) and the PROSE model with only the data modality (i.e. no symbol encoder/decoder or fusion structures).

The comparison tests are run using varying numbers of input sensors. For consistency, noise on the data samples is not included in this test, although the symbolic inputs do have unknown

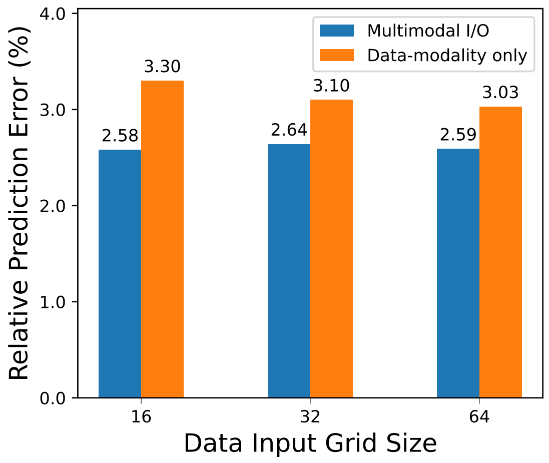


Figure 3: **Comparing the PROSE model with multimodal input/output and the PROSE model with only the data modality.** The models are trained with different data input lengths for 60 epochs. The relative prediction errors are computed on the same output grid.

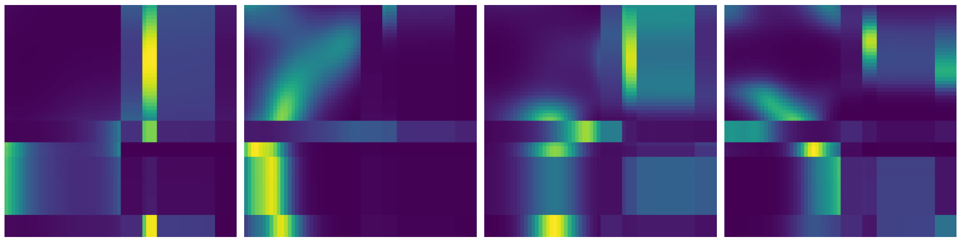


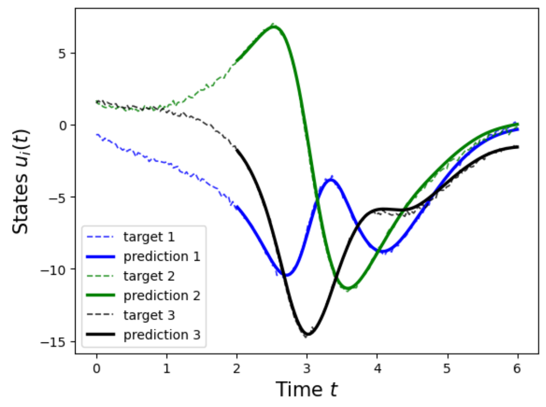
Figure 4: **Sampled attention maps of feature fusion layers.** For each map, non-zero values in the upper left and bottom right corner represent in-modality interactions and non-zero values in the upper right and bottom left blocks represent cross-modality interactions. Other maps are presented in Appendix C.

coefficients and terms added/removed. As shown in Figure 3, the PROSE model with multimodal input/output consistently outperforms the data-modality-only model, demonstrating performance gains through equation embedding. Notably, we do not observe any degradation in the full PROSE model's performance when reducing the number of input sensors, whereas the data-modality-only model's performance declines as sensors are removed from the input function. This showcases the value of the symbol modality in supplying additional information for enhancing data prediction.

In Figure 4, we plot 4 (out of the $64 = 8$ layers $\times$ 8 heads) attention maps corresponding to the Feature Fusion layers on one four-wing attractor example (see Appendix B). This uses the full PROSE model with multimodal input/output and with a data input grid size 32. The non-zero values (which appear as the yellow/green pixels) indicate the connections between the features. More importantly, the non-zero values in the bottom-left and upper-right blocks indicate a non-trivial cross-modality interaction. Together with the improved relative error shown in Figure 3, we see the overall improvements using our multimodal framework.

Output Example. In Figure 5, we display a typical PROSE output from the "Unknown (3D)" experiment in Table 2. Each curve is one trajectory of one state variable $u_i(t)$ for $i = 1, 2, 3$.

The target solution curves (with noise) are the dashed lines (only up to $t = 2$ is seen during testing) and the predicted solution curves are the solid lines. We display the target equation and the generated equation, which is exact with respect to the terms generated and accurate up to two digits (noting that the mantissa has length three).



Target:
$$\begin{cases} u_1' = -0.327u_1 - u_2 - u_3 - 0.25u_2^2 \\ u_2' = -0.327u_2 - u_3 - u_1 - 0.25u_3^2 \\ u_3' = -0.327u_3 - u_1 - u_2 - 0.25u_1^2 \end{cases}$$

Generated:
$$\begin{cases} u_1' = -0.32u_1 - u_2 - u_3 - 0.25u_2^2 \\ u_2' = -0.32u_2 - u_3 - u_1 - 0.25u_3^2 \\ u_3' = -0.32u_3 - u_1 - u_2 - 0.25u_1^2 \end{cases}$$

Relative Prediction Error: 3.29%
Relative Expression Error: 0.36%

Figure 5: **An example of PROSE's outputs.** Target solution curves are dashed lines and predicted solution curves are solid lines. The input is the data up to $t = 2$. The numbers in the legend refer to the coordinate of the state variable $u_i(t)$ for $i = 1, 2, 3$. The target and PROSE generated equations are displayed.

5 Discussion

The PROSE network is developed for model and multi-operator discovery. The network architecture utilizes hierarchical transformers to incorporate the data and embedded symbols in a symbiotic way. We show that the learned symbolic expression helps reduce the prediction error and provides further insights into the dataset. Experiments show that the generated symbolic expressions are mathematical equations with validity of $> 99.9\%$ on in-distribution tests and $> 97.89\%$ on out-of-distribution tests, and with numerical error of about 2% (in terms of relative L^2 norm). This shows that the network is able to generate ODE models that correctly represent the dataset and does so by incorporating information from other similar ODEs.

The symbolic expression and data fusion yield a scientifically relevant multimodal formulation. In particular, the expressions provide alternative representation for the dataset and its predicted values, enabling the extraction of more refined information such as conserved quantities, stationary points, bifurcation regimes, hidden symmetries, and more. Additionally, since the symbolic expressions are valid functions, they can be used for evaluation and thus lead to alternative predictive algorithms (i.e. simulating the ODE). One future direction is the construction of a PROSE approach for nonlinear partial differential equations with spatio-temporal queries.

Acknowledgments

Y. Liu was supported in part by an NSF CAREER Award DMS-2331100. Z. Zhang was supported in part by NSF DMS-2331033. H. Schaeffer was supported in part by AFOSR MURI FA9550-21-1-0084, NSF DMS-2331033, and an NSF CAREER Award DMS-2331100.

References

[1] Shady E Ahmed and Panos Stinis. A multifidelity deep operator network approach to closure for multiscale systems. *Computer Methods in Applied Mechanics and Engineering*, 414:116161, 2023.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*, 2020.

[6] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[7] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[8] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[9] François Charton. Linear algebra with transformers, 2022.

[10] Tianping Chen and Hong Chen. Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Transactions on Neural networks*, 4(6):910–918, 1993.

[11] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

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

[13] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[14] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[15] Stéphane d'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and François Charton. Deep symbolic regression for recurrent sequences, 2022.

[16] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics, 2016.

[17] Yalchin Efendiev, Wing Tat Leung, Guang Lin, and Zecheng Zhang. Efficient hybrid explicit-implicit learning for multiscale problems. *Journal of Computational Physics*, 467:111326, 2022.

[18] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2020.

[19] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[20] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[21] Moritz Hoffmann, Christoph Fröhner, and Frank Noé. Reactive sindy: Discovering governing reactions from concentration data. *The Journal of chemical physics*, 150(2), 2019.

[22] Zhongyi Jiang, Chunmei Wang, and Haizhao Yang. Finite expression methods for discovering physical laws from data. *arXiv preprint arXiv:2305.08342*, 2023.

[23] Pengzhan Jin, Shuai Meng, and Lu Lu. Mionet: Learning multiple-input operators via tensor product. *SIAM Journal on Scientific Computing*, 44(6):A3490–A3514, 2022.

[24] Lee K Jones. A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training. *The annals of Statistics*, pages 608–613, 1992.

[25] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[26] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and Francois Charton. End-to-end symbolic regression with transformers. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[27] 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.

[28] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR, 2021.

[29] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

[30] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

[31] Wing Tat Leung, Guang Lin, and Zecheng Zhang. Nh-pinn: Neural homogenization-based physics-informed neural network for multiscale problems. *Journal of Computational Physics*, 470:111539, 2022.

[32] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[33] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13401–13412, 2021.

[34] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[35] Senwei Liang and Haizhao Yang. Finite expression method for solving high-dimensional partial differential equations. *arXiv preprint arXiv:2206.10121*, 2022.

[36] Guang Lin, Christian Moya, and Zecheng Zhang. Accelerated replica exchange stochastic gradient langevin diffusion enhanced bayesian deeponet for solving noisy parametric pdes. *arXiv preprint arXiv:2111.02484*, 2021.

[37] Guang Lin, Christian Moya, and Zecheng Zhang. On learning the dynamical response of nonlinear control systems with deep operator networks. *arXiv preprint arXiv:2206.06536*, 2022.

[38] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586, 2021.

[39] Yuxuan Liu, Scott G McCalla, and Hayden Schaeffer. Random feature models for learning interacting dynamical systems. *Proceedings of the Royal Society A*, 479(2275):20220835, 2023.

[40] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[41] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[42] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[43] Lu Lu, Raphaël Pestourie, Steven G Johnson, and Giuseppe Romano. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 4(2):023210, 2022.

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

[45] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi,

Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[46] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[47] Benjamin Peherstorfer and Karen Willcox. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[48] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[49] Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019.

[50] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

[51] Hayden Schaeffer and Scott G McCalla. Sparse model selection via integral terms. *Physical Review E*, 96(2):023302, 2017.

[52] Hayden Schaeffer, Giang Tran, and Rachel Ward. Learning dynamical systems and bifurcation via group sparsity. *arXiv preprint arXiv:1709.01558*, 2017.

[53] Hayden Schaeffer, Giang Tran, and Rachel Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.

[54] Hayden Schaeffer, Giang Tran, Rachel Ward, and Linan Zhang. Extracting structured dynamical systems using sparse optimization with very few samples. *Multiscale Modeling & Simulation*, 18(4):1435–1461, 2020.

[55] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[56] Daniel E Shea, Steven L Brunton, and J Nathan Kutz. Sindy-bvp: Sparse identification of nonlinear dynamics for boundary value problems. *Physical Review Research*, 3(2):023255, 2021.

[57] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7464–7473, 2019.

[58] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics, 2015.

[59] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[60] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2019, page 6558. NIH Public Access, 2019.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[62] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[63] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[64] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[65] Peng Xu, Xiatian Zhu, and David A Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[66] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

[67] Liu Yang, Tingwei Meng, Siting Liu, and Stanley J Osher. Prompting in-context operator learning with sensor data, equations, and natural language. *arXiv preprint arXiv:2308.05061*, 2023.

[68] Christopher Yeh, Jing Yu, Yuanyuan Shi, and Adam Wierman. Online learning for robust voltage control under uncertain grid topology. *arXiv preprint arXiv:2306.16674*, 2023.

[69] Zecheng Zhang, Wing Tat Leung, and Hayden Schaeffer. Belnet: basis enhanced learning, a mesh-free neural operator. *Proceedings of the Royal Society A*, 479(2276):20230043, 2023.

[70] Zecheng Zhang, Christian Moya, Wing Tat Leung, Guang Lin, and Hayden Schaeffer. Bayesian deep operator learning for homogenized to fine-scale maps for multiscale pde. *arXiv preprint arXiv:2308.14188*, 2023.

A Experiment Setup

Training A standard cross-entropy loss $\mathcal{L}_s$ is used for the symbolic outputs. While it is possible to simplify and standardize equations with `SymPy` [45], [15] showed that for their symbolic regression task, such simplification decreases training loss but not testing loss, thus we did not include it in our experiments.

Relative squared error $\mathcal{L}_d$ is used for the data predictions. In comparison to the standard mean squared error, the relative squared error makes the learning process more uniform across different types of ODE systems, as solution curves of different systems may have very different value ranges.

The data loss $\mathcal{L}_d$ and symbol loss $\mathcal{L}_s$ are combined to form the final loss function $\mathcal{L} = \alpha\mathcal{L}_d + \beta\mathcal{L}_s$, where the weights α, β are hyperparameters. Unless otherwise specified, the models are trained using the AdamW optimizer for 80 epochs where each epoch is 2,000 steps. On 2 NVIDIA GeForce RTX 4090 GPUs with 24 GB memory each, the training takes about 19 hours.

Hyperparameters The model hyperparameters are summarized in Table 5, and the optimizer hyperparameters are summarized in Table 6.

Table 5: **Model hyperparameters.** FFN means feedforward network.

Hidden dimension for attention	512	Hidden dimension for FFNs	2048
Number of attention heads	8	Fusion attention layers	8
Data encoder attention layers	2	Data decoder attention layers	8
Symbol encoder attention layers	4	Symbol decoder attention layers	8

Table 6: **Optimizer hyperparameters.**

Learning rate	10^{-4}	Weight decay	10^{-4}
Scheduler	Inverse square root	Warmup steps	10% of total steps
Batch size per GPU	256	Gradient norm clip	1.0
Data loss weight α	6.0	Symbol loss weight β	1.0

B Chaotic and MultiScale ODE Dataset

In this section, we provide the details of all ODE systems. We also include the parameters of interest.

Thomas' cyclically symmetric attractor

$$\begin{cases} u_1' &= \sin(u_2) - bu_1 \\ u_2' &= \sin(u_3) - bu_2 \\ u_3' &= \sin(u_1) - bu_3 \end{cases} \qquad b = 0.17$$

Lorenz 3D system

$$\begin{cases} u_1' &= \sigma(u_2 - u_1) \\ u_2' &= u_1(\rho - u_3) - u_2 \\ u_3' &= u_1u_2 - \beta u_3 \end{cases} \qquad \begin{cases} \sigma = 10 \\ \beta = 8/3 \\ \rho = 28 \end{cases}$$

Aizawa attractor

$$\begin{cases} u_1' = (u_3 - b)u_1 - du_2 \\ u_2' = du_1 + (u_3 - b)u_2 \\ u_3' = c + au_3 - u_3^3/3 - u_1^2 + fu_3u_1^3 \end{cases} \qquad \begin{cases} a = 0.95 \\ b = 0.7 \\ c = 0.6 \\ d = 3.5 \\ e = 0.25 \\ f = 0.1 \end{cases}$$

Chen-Lee attractor

$$\begin{cases} u_1' = au_1 - u_2u_3 \\ u_2' = -10u_2 + u_1u_3 \\ u_3' = du_3 + u_1u_2/3 \end{cases} \qquad \begin{cases} a = 5 \\ d = -0.38 \end{cases}$$

Dadras attractor

$$\begin{cases} u_1' = u_2/2 - au_1 + bu_2u_3 \\ u_2' = cu_2 - u_1u_3/2 + u_3/2 \\ u_3' = du_1u_2 - eu_3 \end{cases} \qquad \begin{cases} a = 1.25 \\ b = 1.15 \\ c = 0.75 \\ d = 0.8 \\ e = 4 \end{cases}$$

Rössler attractor

$$\begin{cases} u_1' = -u_2 - u_3 \\ u_2' = u_1 + au_2 \\ u_3' = b + u_3(u_1 - c) \end{cases} \qquad \begin{cases} a = 0.1 \\ b = 0.1 \\ c = 14 \end{cases}$$

Halvorsen attractor

$$\begin{cases} u_1' = au_1 - u_2 - u_3 - u_2^2/4 \\ u_2' = au_2 - u_3 - u_1 - u_3^2/4 \\ u_3' = au_3 - u_1 - u_2 - u_1^2/4 \end{cases} \qquad a = -0.35$$

Rabinovich–Fabrikant equation

$$\begin{cases} u_1' = u_2(u_3 - 1 + u_1^2) + \gamma u_1 \\ u_2' = u_1(3u_3 + 1 - u_1^2) + \gamma u_2 \\ u_3' = -2u_3(\alpha + u_1u_2) \end{cases} \qquad \begin{cases} \alpha = 0.98 \\ \gamma = 0.1 \end{cases}$$

Sprott B attractor

$$\begin{cases} u_1' = au_2u_3 \\ u_2' = u_1 - bu_2 \\ u_3' = c - u_1u_2 \end{cases} \qquad \begin{cases} a = 0.4 \\ b = 1.2 \\ c = 1 \end{cases}$$

Sprott-Linz F attractor

$$\begin{cases} u_1' = u_2 + u_3 \\ u_2' = -u_1 + au_2 \\ u_3' = u_1^2 - u_3 \end{cases} \qquad a = 0.5$$

Four-wing chaotic attractor

$$\begin{cases} u_1' = au_1 + u_2u_3 \\ u_2' = bu_1 + cu_2 - u_1u_3 \\ u_3' = -u_3 - u_1u_2 \end{cases} \qquad \begin{cases} a = 0.2 \\ b = 0.01 \\ c = -0.4 \end{cases}$$

Duffing equation

$$\begin{cases} u_1' = 1 \\ u_2' = u_3 \\ u_3' = -\delta u_3 - \alpha u_2 - \beta u_2^3 + \gamma \cos(\omega u_1) \end{cases} \qquad \begin{cases} \alpha = 1 \\ \beta = 5 \\ \gamma = 8 \\ \delta = 0.02 \\ \omega = 0.5 \end{cases}$$

Lorenz 96 system

$$\begin{cases} u_i' = (u_{i+1} - u_{i-2})u_{i-1} - u_i + F, \ i = 1, \dots, N \\ u_{-1} = u_{N-1}, \ u_0 = u_N, \ u_{N+1} = u_0 \end{cases} \qquad F = 8$$

Double Pendulum

$$\begin{cases} u_1' = u_3 \\ u_2' = u_4 \\ u_3' = \frac{-3g/l \sin(u_1) - g/l \sin(u_1 - 2u_2) - 2\sin(u_1 - u_2)(u_4^2 + u_3^2 \cos(u_1 - u_2))}{3 - \cos(2(u_1 - u_2))} \\ u_4' = \frac{\sin(u_1 - u_2)(4u_3^2 + 4g/l \cos(u_1) + u_4^2 \cos(u_1 - u_2))}{3 - \cos(2(u_1 - u_2))} \end{cases} \qquad \begin{cases} g = 9.81 \\ l = 1 \end{cases}$$

The initial conditions for the ODE systems are sampled uniformly from the hypercube $[-2, 2]^d$ where d is the dimension of the system. The ODE systems are solved on the interval $[0, 6]$ using BDF method with absolute tolerance 10^{-6} and relative tolerance 10^{-5}. Unless otherwise specified, the data part contains function values at 192 uniform grid points in the time interval $[0, 6]$, where the first 64 points in the interval $[0, 2]$ are used as data input points, and the last 128 points in the interval $[2, 6]$ are used as data labels. 2% Gaussian observation noise is added to the data samples. More precisely, if $\boldsymbol{u}$ is the underlying true equation values, the observed value is $\tilde{\boldsymbol{u}} = \boldsymbol{u} + \sigma\boldsymbol{\eta}$ where $\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and σ is chosen such that the signal-to-noise ratio $\frac{\sigma||\boldsymbol{\eta}||_2}{||\boldsymbol{u}||_2}$ is 2%.

C Visualizatons

Figure 6 shows the training and validation loss curves for experiments "Unknown (3D)" and "Skeleton" (described in Table 2). Figure 7 contains the attention architecture details. Figure 8 shows the full attention maps for one four-wing attractor example.

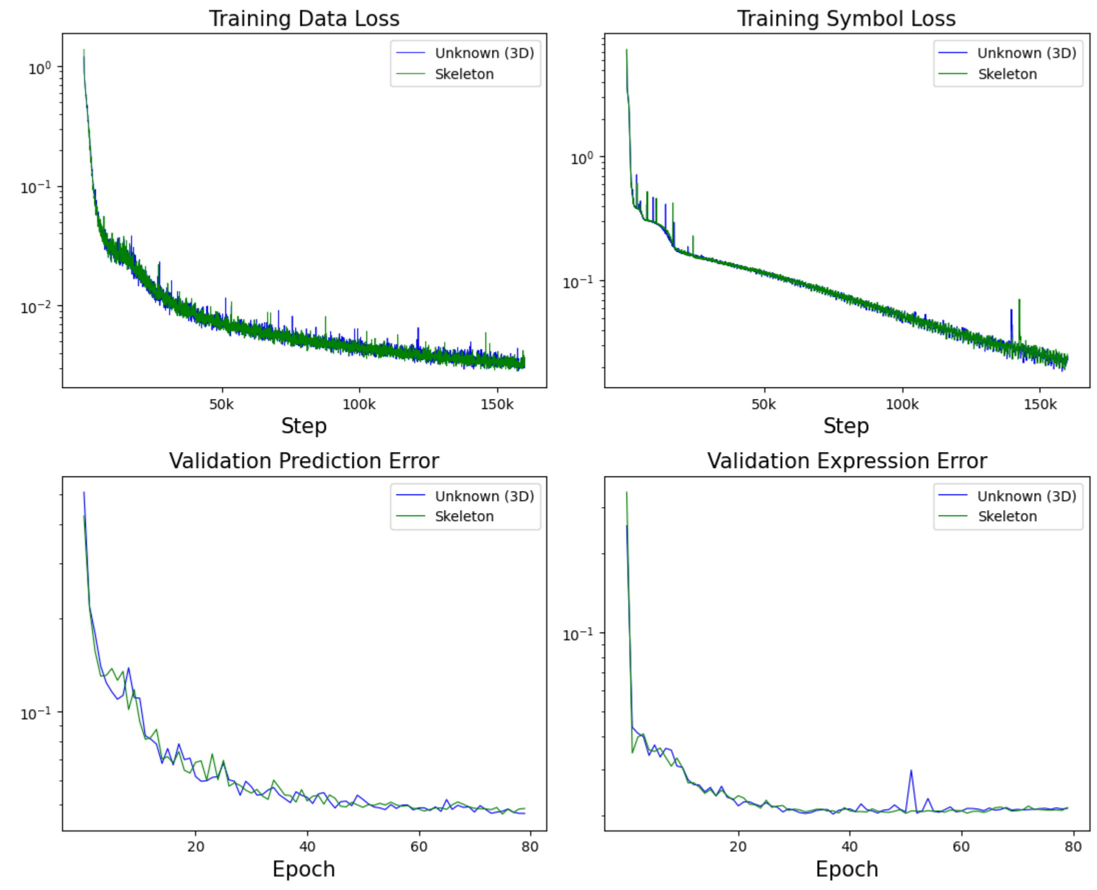
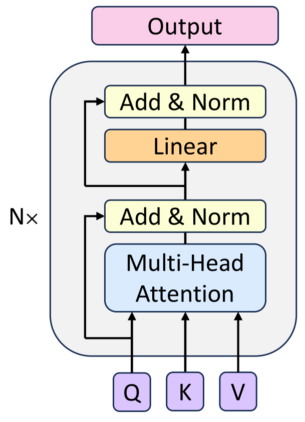
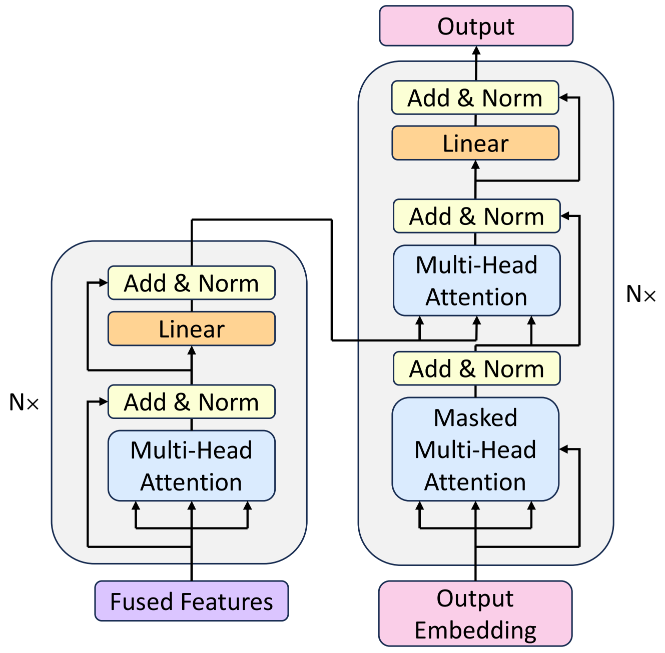


Figure 6: **Example training and validation loss curves.**



(a) Cross-attention



(b) Encoder-decoder attention

Figure 7: **Attention block details.** Self-attention is a special case of cross-attention with the same source.

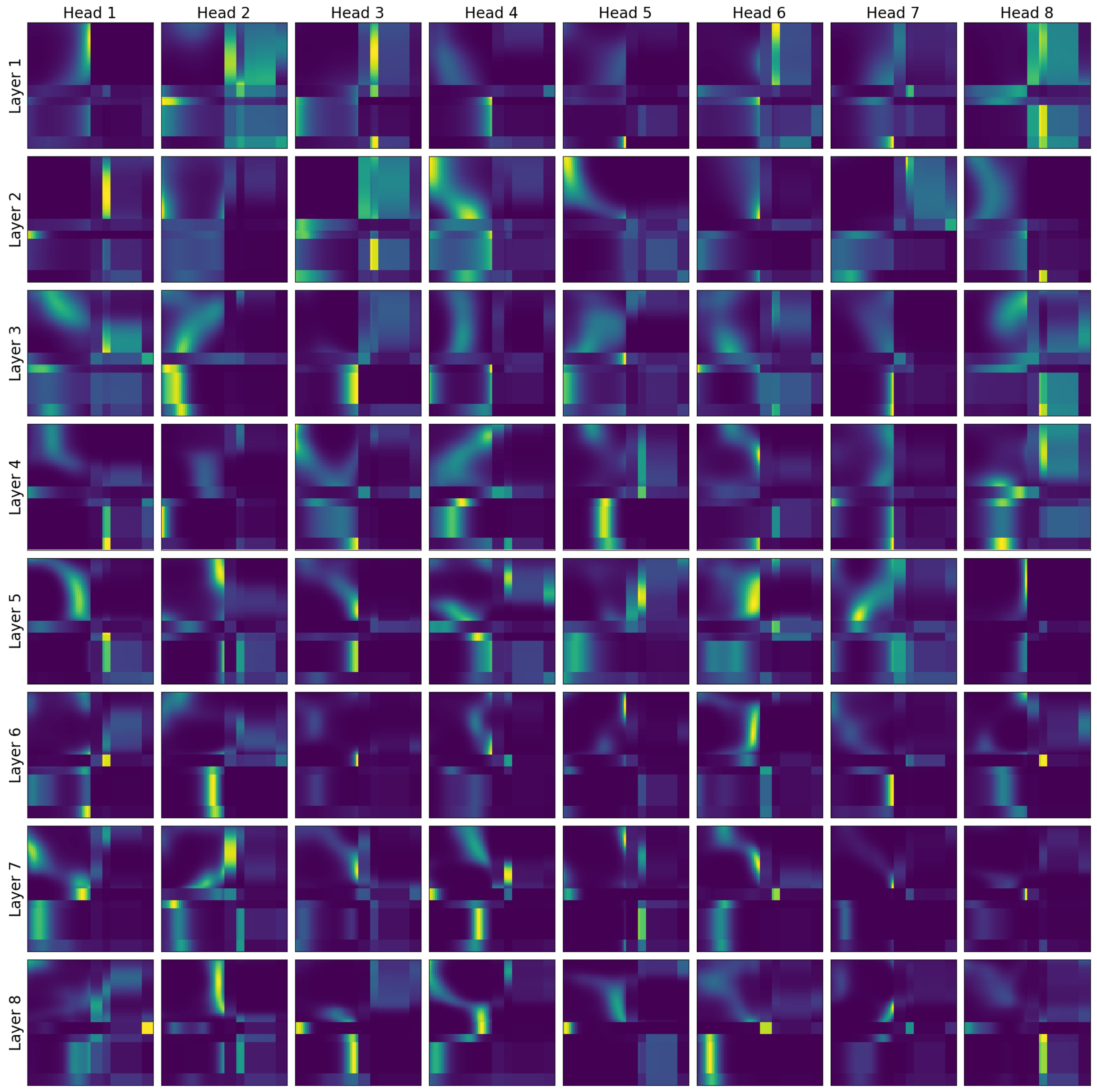


Figure 8: **Attention maps of 8 Feature Fusion layers for a four-wing attractor example.**