NIERT: Accurate Numerical Interpolation Through Unifying Scattered Data Representations Using Transformer Encoder

Shizhe Ding, Boyang Xia, and Milong Ren

Abstract—Interpolation for scattered data is a classical problem in numerical analysis, with a long history of theoretical and practical contributions. Recent advances have utilized deep neural networks to construct interpolators, exhibiting excellent and generalizable performance. However, they still fall short in two aspects: 1) inadequate representation learning, resulting from separate embeddings of observed and target points in popular encoder-decoder frameworks and 2) limited generalization power, caused by overlooking prior interpolation knowledge shared across different domains. To overcome these limitations, we present a Numerical Interpolation approach using Encoder Representation of Transformers (called NIERT). On one hand, NIERT utilizes an encoder-only framework rather than the encoder-decoder structure. This way, NIERT can embed observed and target points into a unified encoder representation space, thus effectively exploiting the correlations among them and obtaining more precise representations. On the other hand, we propose to pre-train NIERT on large-scale synthetic mathematical functions to acquire prior interpolation knowledge, and transfer it to multiple interpolation domains with consistent performance gain. On both synthetic and real-world datasets, NIERT outperforms the existing approaches by a large margin, i.e., 4.3~11.7× lower MAE on TFRD subsets, and 1.7/1.8/8.7× lower MSE on Matht/PhysioNet/PTV datasets.

Index Terms—Interpolation algorithm, pre-trained models, scattered data, transformer encoder.

I. INTRODUCTION

SCATTERED data refers to a set of points and their corresponding values, in which the points lack structured relationships beyond their relative positions [1]. Such data commonly arise in a wide range of theoretical and practical scenarios, including solving partial differential equations (PDEs) [1], [2], temperature field reconstruction [3], particle tracking velocimetry [4], and irregularly-sampled time series analysis [5], [6] (Fig. 1(a)). Numerical interpolation is often required for scattered data, which involves estimating values for target points based on the exact values at some observed points. For instance, in temperature field reconstruction for micro-scale electronics, interpolation methods are employed to obtain real-time working environments for electronic components from limited measurements, and inaccurate interpolation can significantly increase the cost of predictive maintenance. Therefore, accurate numerical interpolation approaches are highly desirable.

A large number of approaches have been proposed for interpolating scattered data. Traditional approaches utilize a linear combination of pre-defined basis functions to approximate the target function, as demonstrated in Fig. 1(b) [7]. To adapt to different scenarios, various types of basis functions have been devised. For example, the spline interpolation algorithm uses polynomials of a small degree over subintervals of the approximation domain as basis functions, and the RBF interpolation algorithm uses radial basis functions. Most of these schemes can theoretically guarantee interpolation accuracy when sufficient observed points are available; however, they have also been shown to be ineffective for sparse data points or complex functions [8]. Furthermore, these methods are inherently not learning-based, i.e., they cannot take advantage of data from the same function distribution, which limits their generalizability and interpolation accuracy.

Recent advances have exhibited an alternative strategy that uses neural networks to learn target functions from the given observed points. For example, conditional neural processes (CNPs) [9] and their extensions [10], [11], [12] use neural networks to model the conditional distribution of regression functions given the observed points, and [3] proposed to use Transformer [13] to solve interpolation tasks in temperature field reconstruction. Unlike traditional approaches that directly output explicit target functions, these neural models take the observed points and target points’ locations as input, and estimate...
the values of these target points. Thus, it is reasonable to view this task as a set-to-set task, which motivates all of these models to employ encoder-decoder structures (Fig. 1(c)). The encoder here embeds the observed points to a latent space while the decoder estimates values for target points. Compared with the traditional approaches, these learning-based methods can learn the function distribution as well as the correlation among scattered points from existing datasets and generalize to new interpolation tasks, which significantly improves the interpolation accuracy.

Despite the excellent performance of existing learning-based interpolation models for scattered data, they still fall short in two aspects: 1) overlooking the homogeneity of observed and target points, which leads to inadequate representation learning of scattered points, and 2) lack of prior interpolation knowledge, which leads to limited generalization power. The underlying reasons of these limitations are: First, the encoder-decoder-based models process the observed and target points in the encoder and decoder separately, which keeps their learned representations in different feature spaces. However, the observed points and target points are intrinsically homogeneous, i.e., they are all sampled from the same function. Treating observed and target points separately may hinder the transfer of information from observed points to target points and increase the difficulty of learning the correlations between the observed points and target points. This strategy makes the representation of target points not precise enough, which limits the final interpolation accuracy.

Second, another shortcoming of these learning-based interpolation models is the lack of prior interpolation knowledge. The continuity and smoothness of the target function are always considered in traditional scattered data interpolation algorithms, e.g., using smooth basis functions to obtain smooth target functions [1]. This prior knowledge reflects the general properties of the function distribution, effectively narrowing the potential range of the target function and guaranteeing interpolation accuracy [7]. However, these existing neural interpolators lack such prior knowledge since they learn interpolation knowledge only from a domain-specific scattered dataset. This makes them lack generalization power to various domains.

To address these limitations, we propose a highly-accurate learning-based numerical interpolation approach for scattered data, which can effectively learn precise point representations with high generalization power. The basic ideas of our approach is as follows:

First, to obtain more precise representations of the target points, we propose a novel interpolation model, called NIERT, which uses a transformer-based encoder-only structure rather than popular encode-decoder structures. Different from previous works, NIERT processes both observed and target points in a unified fashion (see Fig. 1(d)). To bridge the gap between target points and observed points, i.e., to address the missing values of target points, we augment target points with learnable mask tokens, which is similar to BERT [14]. Then we feed them into the encoder together with the observed points. This design improves modeling the correlations between these two types of points, which yields more precise representations and improves interpolation interpretability.1

Moreover, to prevent the unexpected interference posed by target points on the observed points, we introduce a specifically designed attention mask technique [13], [15] with a one-sided pattern which computes attention among the observed data points, excluding the influence of target points at each layer. This mechanism not only aligns with the inductive bias for interpolation tasks, ensuring that the interpolation results are permutation invariant concerning the observed points and permutation equivariant concerning the target points, and that the target function solely depends on the observed points, but it also significantly enhances interpolation robustness.

Second, we use the pre-training technique to incorporate the prior interpolation knowledge into our NIERT interpolator, which has been successfully used for representation learning [14], [16]. The key step towards this goal is building a task set for scattered-data interpolation with rich prior interpolation

1The higher “interpretability” means more visually reasonable correlations between observed and target points (See Section V-C for details.)
knowledge. For this purpose, we synthesized a large number of symbolic mathematical functions and generated a new task set by sampling from these functions. Large-scale syntheses of mathematical functions have the potential to cover real-world continuous and smooth functions in many domains at low cost. With this enhancement, the pre-trained NIERT interpolators can be easily transferred to a variety of domain-specific tasks with a lower learning difficulty and higher interpolation accuracy.

The main contributions of this study are summarized as follows.

- We design a transformer-based encoder-only interpolation framework and embed the observed and target points into a unified feature space, to obtain more accurate target point representations. Also, we propose to use an attention mask mechanism with a one-sided pattern with a strong inductive bias for interpolation tasks, to avoid the interference of one type of points onto the others, which also boosts robustness.

- We leverage the pre-training technique to improve the generalization power of interpolation models. We synthesize a large-scale interpolation task set by generating a massive amount of symbolic mathematical functions. To the best of our knowledge, this study is the first work to propose the pre-trained models for scatter-data interpolation.

- Our approach significantly outperforms both state-of-the-art learning-based approaches and traditional approaches on 4 synthetic and real-world datasets in terms of both interpolation accuracy and interpretability, which shows the potential of our approach in a wide range of application fields.

The paper is structured as follows: Section II provides a literature review. Section III presents our approach. Section IV elaborates on the experimental setup. Section IV reports experimental results and analysis. Section VI discusses the connections between our approach and traditional interpolation methods. Finally, Section VII draws conclusions.

II. RELATED WORK

A. Traditional Interpolation Approaches for Scattered Data

Traditional interpolation approaches for scattered data construct interpolation functions through combining explicitly predefined basis functions. The representative approaches include Lagrange interpolation, Newton interpolation [7], B-spline interpolation [17], Shepard’s method [18], Kriging [19], and radial basis function interpolation (RBF) [20], [21]. Among these approaches, the classical Lagrange interpolation, Newton interpolation and B-splines interpolation are usually used for univariate interpolation. [22] proposed a high-order multivariate approximation scheme for scattered data sets, in which approximation error is represented using Taylor expansions at data points, and basis functions are determined by minimizing the approximation error.

B. Neural Network-Based Interpolation Approaches

Equipped with deep neural networks, data-driven interpolation and reconstruction methods show great advantages and potential. For instance, convolutional neural networks (CNNs) have been applied in the interpolation tasks of single image super-resolution [23], [24], and recurrent neural networks (RNNs) and Transformers have been used for interpolation of sequences like time series data [6], [25].

Recently, [9] proposed to model the conditional distribution of regression functions given observed points. The proposed approach, called conditional neural processes (CNPs), has shown increased estimation accuracy and generalizing ability. A variety of studies have been performed to enhance CNPs: [10] designed attentive neural processes (ANPs), which shows improved accuracy, [12] leveraged Bayesian last layer (BLL) [26] for faster training and better prediction. In addition, the bootstrap technique was also employed for further improvement [11]. To solve the interpolation task in 2D temperature field reconstruction, [3] proposed a Transformer-based approach, referred to as TFR-transformer, which can also be applied to solve interpolation tasks for scattered data with higher dimensions.

C. Transformer-Based Language/Image Models and the Pre-Training Technique

Transformer-Based Language/Image Models: The design of NIERT is also inspired by the recent advances in large language/images models [14], [16], [27], [28], [29], [30], [31]. Among these, the Transformer encoder-only models such as BERT [14] and BEIT [16] leverage masked token to reconstruct missing data from their context and to effectively learn representations of language and images. Transformer decoder-only models like the GPT series [27], [28], [29] are built upon masked self-attention modules, which prevents future tokens from affecting the current token’s encoding, thus enhancing the generative capabilities.

Importantly, while our encoder-only architecture bears similarities to BERT and BEIT, it significantly differs in terms of model functionality and design. Functionally, our model focuses on the task of scattered data interpolation, outputting interpolated results for target points, distinct from BERT, which produces representations of text. In design, we utilize a one-sided attention mask to establish a unidirectional flow of information from source to target, preventing representation degradation. Additionally, our proposed ‘one-sided attention mask’ shares some similarities with masked self-attention [13], [27], but their objectives and implementations are distinct. Unlike masked self-attention which deals with generic tokens in Transformer decoder or GPT, ‘one-sided attention mask’ is specifically designed for interpolation tasks, in which ‘tokens’ can be split into two categories: observed and target points. The mechanism negates the influence of any target point on the encodings of other points, leading to a rectangular masking pattern (as shown in Fig. 2), which is unlike the lower-triangular mask pattern in masked self-attention.

Pre-Training Techniques: As a way of introducing prior knowledge, large-scale pre-training allows language or visual models to acquire more general text or image representation capabilities, which assists the learning process on downstream tasks and significantly improves their performance [14], [27].
addition, the pre-trained models for symbolic regression were designed to learn the map from scattered data to corresponding symbolic formulas [32], [33]. Here, NIERT adopts pre-training technique to acquire strong generalization power to reconstruct the missing values of scattered points.

### III. Method

#### A. Overview of NIERT Approach

In the study, we focus on the interpolation task that can be formally described as follows: We are given $n$ observed points with known values $O = \{(x_i, y_i)\}_{i=1}^n$, and $m$ target points with values to be determined, denoted as $T = \{(x_i, y_i)\}_{i=n+1}^{n+m}$. Here, $x_i \in \mathbb{R}^{d_x}$ denotes the position of a point, $y_i = f(x_i) \in \mathbb{R}^{d_y}$ denotes the value of a point, and $f: \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$ denotes a function mapping positions to values. The $d_x$ and $d_y$ denote the dimension of a point’s position and the dimension of the value, respectively. The function $f$ is from a function distribution $\mathcal{F}$, which can be explicitly defined using a mathematical formula or implicitly represented using a set of scattered data in the form $(x_i, y_i)$. The goal of the interpolation task is to accurately estimate the values $f(x)$ for each target point $x \in T$ according to the observed points in $O$.

The main element of our NIERT approach is a neural interpolator that learns to estimate values for target points. We train the neural interpolator using a set of interpolation tasks sampled from the function distribution $\mathcal{F}$ (Section III-B and Fig. 2). For each of the sampled interpolation tasks, we mask the values of target points, feed the task to the neural interpolator, and train the interpolator to predict the values of target points and observed points as well. The training objective is to minimize the error between the predicted values and the ground-truth values of both target and observed points. The trained interpolator can be used to interpolate values of target points for the interpolation tasks from the distribution $\mathcal{F}$. In addition, to enable NIERT with generalization power to various interpolation domains, we pre-train NIERT on a large-scale interpolation task set Mathit, which we generate by sampling on synthesized mathematical symbolic functions (see Section III-C and Fig. 3). In the following sections, we first present the NIERT encoder-only structure, then we introduce the pre-training technique.

#### B. Architecture of the NIERT Interpolator

The neural interpolator in NIERT adopts the Transformer encoder framework; however, to suit the interpolation task, significant modifications and extensions were made in the embedding, Transformer and output layers, which are described in detail below.

**Unified Representations With Masked Tokens:** NIERT embeds both observed points and target points into the unified high-dimensional embedding space. As the position $x$ of a data point and its value $y$ are from different domains, we use two linear modules: $\text{Linear}_x$ embeds the positions while $\text{Linear}_y$ embeds the values.

We feed the points and target points into the same embedding modules $\text{Linear}_x, \text{Linear}_y$. It should be noted that for target points, their values are absent when embedding as they are to be determined. In this case, we use a masked token as a substitute, which is embedded as a trainable parameter $\text{MASK}_y$ as performed in BERT [14]. This way, the interpolator processes both target points and observed points in a unified fashion.

We concatenate the embeddings of position and value of a data point as the point’s embedding, denoted as $h_i^0$, i.e.,

$$
h_i^0 = \begin{cases} 
\text{Linear}_x(x_i), \text{Linear}_y(y_i), & \text{if } (x_i, y_i) \in O \\
\text{Linear}_x(x_i), \text{MASK}_y, & \text{if } x_i \in T
\end{cases}.
$$

**Attention Mechanism With One-Sided Attention Mask:** NIERT feeds the embeddings of the points into a stack of $L$ Transformer layers, producing encodings of these points as result. Through the $l$-th layer, the encoding of the $i$-th point $h_i^l$ are transformed to $h_i^{l+1}$. Each Transformer layer contains two
Fig. 3. The generation process of Mathit pre-training task set and the pre-training and fine-tuning technique for NIERT interpolator. By randomly sampling the expression trees and coefficients, we synthesized a large number of mathematical functions that can represent a sufficiently wide and diverse function distribution. These functions are evaluated numerically and used to sample the Mathit interpolation tasks used for pre-training. The pre-training introduces prior knowledge into the interpolators and improves their generalization power in various real-world interpolation scenarios.

subsequent sub-layers, namely, a multi-head self-attention module, and a point-wise fully-connected network. These sub-layers are interlaced with residual connections and layer normalization between them.

To avoid the unexpected interference of target points on observed points and target points themselves, NIERT imposes one-sided attention mask mechanism on self-attention in the vanilla Transformer layer. As in vanilla self-attention [13], the point embedding $h^l_i$ is first transformed to the query vector $q^l_i$, key vector $k^l_i$ and value vector $v^l_i$ by linear projection matrices $W^l_Q, W^l_K, W^l_V$, respectively. Then, the unnormalized attention score $e_{ij}$ can be computed by scaled dot production of the query $q^l_i$ and key $k^l_j$:

$$e_{ij} = \frac{d_k}{\sqrt{d_k}} \cdot k^l_j, \quad \text{(1)}$$

where $d_k$ is the dimension of the query and key vectors. Unlike the vanilla self-attention layer, we mask out entries of attention weights between target points and all points by replacing these entries with $-\infty$:

$$\hat{e}_{ij} = \begin{cases} 
-\infty, & \text{if } x_j \in T \\
\hat{e}_{ij}, & \text{otherwise}
\end{cases}.$$

Then the attention score can be obtained by softmax normalization:

$$\alpha^l_{ij} = \frac{\exp(\hat{e}_{ij})}{\sum_p \exp(\hat{e}^l_{i,p})}, \quad \text{(2)}$$

In this way, a masked entry $\alpha^l_{ij}$ is equal to 0 when the $j$-th point is a target point, which is also illustrated in Fig. 2. Thus, only the attention connections starting from the observed points to target points and between observed points themselves are reserved, which exhibits ‘one-sided’ patterns. Then the layer output $h^{l+1}_i$ can be computed as follows:

$$h^{l+1}_i = \text{LayerNorm} \left( \tilde{v}^l_i + \text{MLP} \left( \tilde{v}^l_i \right) \right), \quad \text{(3)}$$

where $\tilde{v}^l_i = \text{LayerNorm} \left( v^l_i + \sum_j \alpha^l_{ij} v^l_j \right). \quad \text{(4)}$

Through one-sided attention mask mechanism, for each observed point $(x_j, y_j) \in O$, we maintain $\alpha^l_{ij}$ not masked; thus, NIERT can model the correlation between observed points and target points and the correlation among observed points themselves (Fig. 2). Relatively, the correlation among observed points is easier to learn as these points have known values. In addition, this correlation can be transferred onto the target points, thus promoting learning representations of these data points. In contrast, by forcing the weight $\alpha^l_{ij}$ to be 0 for each target point $j$, we completely avoid the unexpected interference of target points on the other points. This design will eventually lead to more robust representations of the target points and more accurate values will be estimated.

Estimating Values for Target Points: For each target point $i$, we estimate its value $\hat{y}_i$ through feeding its features at the final Transformer layer into a fully connected feed-forward network, i.e.,

$$\hat{y}_i = \text{MLP}_{\text{out}}(h^L_i).$$
We calculate the error between the estimation and the corresponding ground-truth value, and compose the errors for all points into a loss function $\mathcal{L}$ to be minimized.

### C. Pre-Training NIERT Interpolators

We pre-train the NIERT interpolation model to obtain powerful pre-trained interpolators and apply them to various interpolation scenarios (Fig. 3). Since the dimension of the input and output of the function, $d_x, d_y$, which affects the architecture of the interpolator, may be different in various interpolation scenarios, we build a series of pre-trained interpolation models to accommodate scenarios with different function dimensions $(d_x, d_y)$. The pre-trained NIERT in a certain $(d_x, d_y)$ configuration has strong generalization power to different scenarios with the same dimension configuration. For example, the pre-trained NIERT with $(2,1)$ dimension can be transferred into 3 different subsets of 2D temperature field reconstruction, i.e., HSink, ADlet and DSine, which are temperature field data for three quite different types of physical conditions [3].

The key to the pre-training process is the construction of the pre-training task set. We propose a math function interpolation task set, namely, Mathit, for pre-training our model. These tasks for pre-training are sampled from synthesized mathematical functions (see Fig. 3). These mathematical functions can be evaluated numerically with perfect accuracy and are suitable for sampling interpolation tasks to train highly accurate interpolators. The configured synthesis process is flexible to generate functions of different dimensions, which are adequate for training interpolator families. Synthesizing data exhibits significant advantages over collecting data from the real world on large scale and at low costs.

**Synthesis of Mathematical Expressions:** We borrow the work of [32], [34] to randomly synthesize mathematical functions. A mathematical function is synthesized in the form of an expression tree, where the internal nodes are operators and the leaf nodes are independent variables and coefficients. Eight types of operators with different sampling weights are used in the random synthesis (Table I), and each synthesized expression have at most 6 operators. We use these operators excluding ‘+’ or ‘log’, because it should be guaranteed that the synthesized functions are continuous so as to sample legitimate interpolation tasks. Leaf nodes have a probability of 0.8 to be independent variables and a probability of 0.2 to be coefficients. The optional set of independent variables depends on the dimensionality of the scattered point in the task. For example, when the dimension of the scattered points $d_x$ is 3, the independent variables are chosen with equal probability as $x_1, x_2, x_3$. For each dimension of function input $d_x$, we randomly synthesized one million of these expressions with coefficients and saved them for the subsequent sampling of the interpolation task.

### Sampling Pre-Training Interpolation Tasks: At training time, the coefficients of each expression are randomly sampled from $[-2,2]$, such that the functions seen by the model are all different. Once the coefficients in an expression are determined, the expression can be numerically evaluated as a mathematical function. Next, an interpolation task is sampled from each evaluated function. We randomly sample 256 scattered points from within $[-1,1]^d_x$, and compute the values of the function on these points. The function values of the scattered points can be very large or small, which may lead to exploding or vanishing gradients during training, so we normalize the values of each set of scattered points. To construct an interpolation task, we divide these 256 points into $n$ observed points and $256 - n$ target points randomly. Here, we randomly set $n$ to an integer in $[10,50]$, which induces the generalizability of our model to a variable number of observed points. In this way, an interpolation task for pre-training is sampled.

**Dimensional Augmentation Technique:** To synthesize mathematical functions with output dimension greater than 1 ($d_y > 1$), we use the dimensional augmentation technique, which randomly takes $d_y$ functions from the mathematical function set with independent variable dimension $d_x$ and concatenates them together. We do this based on the simple fact that functions of multiple dependent variables can be directly expressed as a combination of multiple functions of one dependent variable. For example, a two-dimensional velocity field can be expressed as $[v_x(x, y), v_y(x, y)]$.

**Pre-Training and Fine-Tuning:** We pre-train this series of NIERT interpolators separately for different $(d_x, d_y)$ configurations. For a specific $(d_x, d_y)$ configuration, the pre-trained NIERT interpolator can be transferred to different interpolation scenarios of the same dimension configuration. After pre-training, we obtain pre-trained NIERT interpolators with general interpolation capabilities that can be fine-tuned on a domain-specific set of interpolation tasks.

### IV. Experiment Setting

In this section, we present the description of datasets used for our experiments, the baselines used for comparison, and the experimental details.

**A. Datasets**

We evaluate our approach on four representative datasets, including two synthetic datasets and two real-world datasets. These datasets are representatives of interpolation tasks in various application fields: Mathit for mathematical function interpolation, TFRD [3] for temperature field reconstruction, PTV [35] for two-dimensional particle tracking velocimetry, PhysioNet [36] for irregularly-sampled time-series data interpolation. We introduce each dataset in detail as follows.

1) **Synthetic dataset I: Mathit**, which denotes the pre-trained dataset we constructed (described in detail in Section III-C), is used to test the performance for mathematical function interpolation. We evaluate the interpolation accuracy of NIERT and existing approaches on functions of various independent variable dimensions.

### Table I

| Operator | $+$ | $\times$ | $-$ | $^3$ | $^2$ | $\exp$ | $\sin$ | $\cos$ |
|----------|-----|---------|-----|-----|-----|--------|--------|--------|
| Sample weight | 10  | 10  | 5   | 4   | 2   | 4      | 4      | 4      |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
In addition to the pre-trained dataset, we synthesized a test set including 12,000 interpolation tasks for each independent variable dimension of functions. To avoid the intersection between training and test sets, we remove the mathematical expressions appearing in both sets.

2) Synthetic dataset II: TFRD is used to test the performance for 2D temperature field reconstruction. It consists of three sub-datasets with large differences: HSink, ADlet, and DSine, each of which simulates the temperature field of mechanical devices using a specific kind of heat generation and boundary conditions [3]. Each reconstruction task in each subset has 200 \times 200 regular grid points that represent the temperature field in a 0.1 m \times 0.1 m square area. Among these points, 37 scattered points have their temperate known and are used as observed points. The other points are used as target points. In each subset, we have a total of 10,000 training instances and 10,000 test instances. We follow to use L_1-form loss function for this dataset as performed in [3] for a fair a comparison.

3) Real-world dataset I: PTV is a scattered data set where the scattered points are particles with velocities in the flow field. PTV is used to test the performance for particle tracking velocimetry, i.e., the two-dimensional velocity field reconstruction from a finite number of observed particles with velocities. These data are extracted from the raw images of the laminar jet experiment scenes taken by [35]. There are 1200 raw frames in total corresponding to 1200 velocity fields, and each frame extracts a set of scattered points with velocity, which have at most close to 6000 points and at least 2000 points. For each set of scattered data, we randomly take 512 points as observed points and the remaining points as target points to construct an interpolation task. We randomly select a quarter of the tasks as the test set and the rest as the training set.

4) Real-world dataset II: PhysioNet excerpted from the PhysioNet Challenge 2012 [36], is used to test the performance of NIERT for irregularly-sampled time series interpolation. This real-world dataset was collected from intensive care unit (ICU) records. Each instance consists of multiple points, each point represents a measurement of a patient at a specific time, and each measurement contains up to 41 physiological indices. Following the study in [25], we randomly divided the points into observed points and target points, then trained and evaluated interpolation models using the acquired interpolation task sets. We set the ratio of observed points at three levels, i.e., 50\%, 70\%, and 90\% for adequate comparisons. We randomly divided the 8,000 instances of PhysioNet into the training set and test set with a ratio of 4 : 1. Since these time series as functions have a dependent variable of dimension 41, to evaluate the effectiveness of the pre-training on this dataset, we additionally pre-trained a NIERT model suitable for such functions using the dimensional augmentation technique. We list the statistics of all the above datasets in Table II. In practice, for each instance in the training set, i.e., N scattered points from a certain function, we randomly select n of them as observed points and the remaining N – n as target points. For the Mathit dataset, we randomly select a fixed range of numbers as the number of observed points (e.g. [5,50] for Mathit dataset). For the real-world PhysioNet dataset, which has a variable number of scattered points for its instances, distributed in the range of [20,190], we, therefore, select observed points from all points at a fixed rate (say 50\%, 70\% or 90\%).

B. Baselines

All baselines can be divided into two categories: 1) traditional interpolation algorithms and 2) learning-based models.

Traditional Interpolation Algorithms: Radial Basis Function (RBF) Interpolation is one of the most commonly-used scattered data interpolation methods. It adopts a specific type of radial basis function on observed points and uses their linear combination to represent the target function. We use the RBF interpolation implementation in SciPy [37] and multiquadric function as the basis function type for the experiments.

MIR is another multivariate interpolation and regression method for scattered data sets proposed by [22]. MIR represents the approximation error with Taylor expansions and minimizes the approximation error to find the basis functions.

Learning Based Models: Conditional Neural Processes (CNPs) proposed by [9] is a neural model composed of MLPs, which is able to learn to predict distributions of functions given observed points.

Attentive Neural Processes (ANPs) [10] leverages the attention mechanism in CNPs and improves the prediction performance.

Bootstrapping Attentive Neural Processes (BANPs) [11] employs the bootstrap technique to further improve the performance of ANPs.

TFR-Transformer, proposed in [3], uses Transformer [13] in 2-dimensional temperature field reconstruction using scattered observations.

RNN-VAE is a VAE-based model where the encoder and decoder are standard RNN models. Gated Recurrent Unit (GRU) [38] module is configured as the recurrent network.
L-ODE-RNN refers to the latent neural ODE model where the encoder is an RNN and the decoder is a neural ODE proposed in [39].

L-ODE-ODE refers to the model where the encoder is an ODE-RNN [40] and the decoder is a neural ODE.

tTAND-Full [25] performs time attention mechanism and Bidirectional RNNs to encode temporal features and interpolate irregular-sampled time series data.

C. Experimental Details

Model Hyper-Parameters: Table III lists the hyper-parameters of NIERT in the experiments on the four representative datasets, including Mathit, TFRD, PIV and PhysioNet. To fairly evaluate the performance of the pre-trained models, the hyperparameters of the pre-trained models we used on the TFRD, PIV and PhysioNet datasets were aligned with those of NIERT. In addition, we configured the TFR-transformer with the same hyperparameter settings for a fair comparison. For hyperparameter settings of other baselines, we follow the settings in the work of [3] and [25].

Evaluation Metrics: For the Mathit, PTV, and PhysioNet datasets, we used the mean square error (MSE) of interpolation on the target points to evaluate the interpolation accuracy of the model. For the PhysioNet dataset, given its small size and significant irregularity, we follow the approach of [25] by repeating each training and evaluation experiment five times using different random seeds. We use the average MSE as the evaluation metric, with the standard deviation also provided. For the TFRD dataset, using MSE as a metric will yield large performance disparities among approaches. Thus, we used the Mean Absolute Error (MAE) on target points for a more preferable metric, aligning with the approach of [3].

Training Details: We employed the Mean Squared Error (MSE) as the loss function for training on the Mathit, PTV, and PhysioNet datasets, and used the Mean Absolute Error (MAE) as the loss function for training on the TFRD dataset. This approach is consistent with their respective evaluation metric settings. For pre-training, we aligned the loss function, be it MSE or MAE, with the one employed in the corresponding training on the downstream dataset. We used Adam for parameter optimization. For pre-training on the Mathit dataset, we sampled mini-batches of size 150 and used a learning rate of $10^{-4}$, no schedules, and trained for 160 epochs. For training or fine-tuning on the TFRD, PTV and PhysioNet dataset, we sampled mini-batches of size 5, 32 and 4 respectively. The models were trained on these three datasets using a learning rate of $10^{-3}$, and the learning rate was decayed by a factor of 0.97 at the end of each epoch.

### Table III

| Parameter name          | Parameter value in experiments on |
|-------------------------|----------------------------------|
|                         | Mathit | TFRD | PIV | PhysioNet |
| $z$’s embedding dimension | $16 \times d_z$ | 32   | 32  | 16        |
| $y$’s embedding dimension | 16     | 16   | 32  | 592       |
| Number of layers        | 6      | 3    | 3   | 3         |
| Number of heads         | 8      | 4    | 4   | 4         |
| Hidden dimension        | 512    | 128  | 128 | 128       |

### Table IV

| Interpolation approach | MSE ($\times 10^{-2}$) on Mathit test set |
|------------------------|------------------------------------------|
|                        | 1D | 2D | 3D | 4D          |
| RBF                    | 215.439 | 347.060 | 443.094 | 327.775 |
| MIR                    | 67.281  | 274.601 | 448.933 | 312.497 |
| CNP                    | 47.176  | 248.868 | 392.548 | 314.311 |
| ANP                    | 34.558  | 160.005 | 206.699 | 164.751 |
| BANP                   | 14.913  | 84.187  | 143.518 | 140.288 |
| TFR-transformer        | 15.556  | 58.569  | 99.986  | 90.579   |
| NIERT                  | 8.964   | 45.319  | 77.664  | 72.025   |

V. RESULTS

In this section, we present experimental results and analysis, including interpolation accuracy on synthetic and real-world datasets, individual case studies of interpolation, interpretability analysis of attentions, and results of ablation experiments. These results demonstrate the effectiveness and superiority of our approach. We present more results in the Supplementary Material.

A. Interpolation Accuracy on Synthetic and Real-World Datasets

For each instance of the test dataset, we applied the trained NIERT and the existing approaches to estimate values for target points. We calculate the mean of errors between estimation and ground truth as interpolation accuracy.

Accuracy on Mathit Datasets: As shown in Table IV, on the 1D Mathit test set, RBF shows the largest interpolation error (MSE: 215.439). MIR, another approach using explicit basis functions, also shows a high interpolation error of 67.281. In contrast, BANP and TFR-transformer, which use neural networks to learn interpolation, show relatively lower errors (MSE: 14.913, 15.556). Compared with these approaches, our NIERT approach achieves the best interpolation accuracy (MSE: 8.964 v.s. 14.913). Table IV also demonstrates the significant advantage of NIERT over the existing approach on the 2D, 3D, and 4D instances.

To examine in depth the interpolation accuracy, we further divide test instances into subsets according to the number of observed points. As shown in Fig. 4(a) and (b), as the number of observed points increases, the interpolation error decreases as expected. In addition, the relative advantages of these approaches vary with the number of observed points, e.g., CNP is better than RBF and MIR initially but finally becomes worse as the number of observed points increases. Among all approaches, NIERT stably shows the best performance over all test subsets, regardless of the number of observed points.

### Accuracy on TFRD, PTV and PhysioNet Datasets

As shown in Table V, CNP, although employing the neural network technique, still performs poorly on TFRD datasets. In contrast, NIERT achieves much lower interpolation error (23,519 v.s. 204,351, 34739 v.s. 91.782 and 8785 v.s. 92.456 on HSink, ADlet and DSink subsets, respectively), which are also significantly lower than ANP, BANP and TFR-transformer. Moreover,
Fig. 4. The relationship between the interpolation accuracy and the number of observed points. Here we use the instances in the Mathit-1D and Mathit-2D test dataset as representatives.

Table V

| Interpolation approach | MAE ($\times 10^{-3}$) on TFRD test set |
|------------------------|----------------------------------------|
|                        | HSink       | ADlet     | DSine     |
| CNP                    | 204.351     | 91.782    | 92.456    |
| ANP                    | 164.491     | 54.684    | 58.589    |
| BANP                   | 59.728      | 28.671    | 19.107    |
| TFR-transformer        | 64.987      | 27.074    | 29.961    |
| NIERT                  | 23.519      | 3.473     | 8.785     |
| NIERT w/ pretraining  | 13.943      | 1.780     | 4.909     |

When enhanced with the pre-training technique, NIERT can consistently further decrease its interpolation MAE by 35%, 45% and 44% relatively on these 3 subsets, respectively, which verifies the strong generalization power of our pre-trained NIERT interpolator.

Table VI shows that NIERT also outperforms existing methods on the PTV dataset, for example, the average MSE ($\times 10^{-3}$) of NIERT is significantly lower than all other methods (1.954 vs. 17.125). Similarly, the pre-trained NIERT shows better performance. This demonstrates the effectiveness of our NIERT and pre-training method in the 2D particle tracking velocimetry scenario.

Table VII suggests that on the PhysioNet dataset for irregularly-sampled time series interpolation, NIERT also outperforms the existing approaches, e.g., when controlling the ratio of observed points to be 50%, the interpolation error of NIERT significantly lower than other approaches (2.868 vs. 4.139). This superiority persists across different observed point ratio settings.

It is worth noting that the relative improvement achieved by pre-trained NIERT on the PhysioNet dataset is somewhat smaller compared to its performance gains on PTV or TFRD datasets. This discrepancy can be attributed to the substantial dissimilarities between the PhysioNet dataset and the pre-training dataset. The PhysioNet data comprises 41-variable time series data, while the pre-training data consists of combinations of 41 random Mathit-1D functions. Modeling the data distribution of 41 fixed variables using numerous random function combinations can be challenging. Nevertheless, we still observe consistent improvements across various settings of the observed data point ratio, underscoring the effectiveness of the pre-training technique.

Taken together, these results demonstrate the power of NIERT for numerical interpolation in multiple application fields, including mathematical function interpolation, temperature field reconstruction, particle tracking velocimetry, and interpolating irregularly-sampled time-series data. These results also show that the proposed pre-training technique can give NIERT interpolators strong generalization capabilities.

B. Case Studies of Interpolation Results

To further understand the advantages of NIERT, we carried out case studies through visualizing the observed points, the reconstructed interpolation functions and the interpolation errors in this subsection. We have placed an example of Mathit-2D and an example of TFRD-ADlet in Figs. 5 and 6, respectively. More visualized cases are provided in the Supplementary material.

Fig. 5 shows a 2D instance in the Mathit test set. As illustrated, RBF performs poorly in the application scenario with sparse observed data. In addition, MIR and ANP, cannot accurately predict...
values for the target points that fall out of the range restricted by observed points. The CNP approach can only learn the rough trend stated by the observed points, thus leading to significant errors. In contrast, BANP, TFR-transformer and NIERT can accurately estimate values for target points within a considerably large range, and compared with BANP and TFR-transformer, NIERT can produce more accurate results.

Fig. 6 shows an instance of temperature field reconstruction extracted from TFRD-ADlet. From this figure, we can observe that NIERT and pre-trained NIERT have significantly lower interpolation errors over the entire region than existing methods. When using the pre-training technique, NIERT further improves its interpolation accuracy in the whole area (with fewer light-colored blobs than non-pre-trained NIERT).

C. Interpretability Analysis

To analyze the interpretability, we visualized the attention scores in the last attention layer, i.e., $\alpha_{ij}$ in (4), of each observed point to all target points. These the attention scores indicate the correlation between observed and target points. Here, we take a Mathit-2D interpolation task and set the target points to be points from a $128 \times 128$ uniform grid covering the $[-1, 1]^2$ domain, such that these attention weights provide an intuitive description of the contribution to interpolation by each observed point.

As shown in Fig. 7, we compared the attention weights from the final attention layer of NIERT and TFR-transformer. As a reference, we also present the Voronoi polygons [41] of the observed points. The contributions by the other 18 observed points are shown in Section 1.3 in the Supplementary material.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 8. The robustness of NIERT to the number of target points. Here, NIERT uses three attention mechanisms trained on the Mathit-1D dataset. Specifically, the proposed “one-sided attention mask” excludes attention computation from target points to all points. The “vanilla self-attention” does not mask any attention computation. While the “one-sided attention mask + target points’ correlation” builds upon one-sided attention mask by further incorporating attention calculation among target points.

D. Ablation Study

The Effects of One-Sided Attention Mask Mechanism: For a specific interpolation task, the interpolation function is determined by the observed points only. To investigate the effects of one-sided attention mask mechanism in avoiding interference of target points, we evaluated NIERT on the test sets with various numbers of target points. Here, we compared NIERT with its two variants, one using vanilla self-attention, and the other using our one-sided attention mask with the target points’ correlation added. Both of them were trained using the same training sets (the number of target points varies within [206, 246]).

As illustrated by Fig. 8, the two NIERT variants show poor performance for the tasks with few target points, say less than 64 target points or more than 768 target points. This stems from the influence of target points when encoding the representation of scattered points, resulting in learned interpolators reliant on the distribution of target points set during training. Consequently, interpolation accuracy deteriorates significantly when evaluated on tasks with substantially smaller or larger target set sizes compared to the training tasks. In contrast, NIERT, which uses our one-sided attention mask, always performs stably without significant changes in accuracy. The results clearly demonstrate that the one-sided attention mechanism allows NIERT to be free from the unexpected effects of the target points, which enhances the robustness of interpolation.

The Effects of Pre-Training Technique: To investigate the effects of the pre-training technique, we show in Fig. 9 the training process of two versions of NIERT, one without the pre-training technique, and the other enhanced with pre-training. As depicted by Fig. 9(a), on TFRD-ADlet dataset, even only after the first epoch, the pre-trained NIERT shows a sufficiently high interpolation accuracy, which is comparable with the fully-trained BANP and TFR-transformer. On PIV dataset (Fig. 9(b)), the pre-trained NIERT has exceeded the final interpolation accuracy of all existing baselines even after the first epoch of fine-tuning. Moreover, the pre-trained NIERT ends up with the smallest interpolation errors on both datasets, which are nearly half of the error of the NIERT without pre-training.

These results suggest that the pre-training technique gives the NIERT interpolator powerful interpolation generalization capabilities. On one hand, it significantly accelerates the convergence and reduces the cost of transfer learning. On the other hand, it improves the final interpolation performance of NIERT.

Ablations on Distribution of Pre-Training Functions: We conducted further investigations into the influence of function distribution in the pre-training dataset, Mathit, on the generalization ability of pre-trained NIERT. Specifically, we adjusted the function distribution by modifying the operators used in symbolic functions, which are pivotal in controlling function distribution during dataset synthesis.

The default operators in Mathit (Table I) include basic operators {+, ·, −}, transcendental function operators {sin, cos, exp}, and power function operators {2, 3}. Basic operators build symbolic expression functions, while power and transcendental
function operators increase diversity by introducing polynomial signals and signals with periodicity and trends, respectively. We created three Mathit variants, each of equivalent scale to Mathit, with each excluding either transcendental function operators, power function operators, or both during the dataset synthesis.

We assessed the generalization of NIER pre-trained on these Mathit variants by fine-tuning the pre-trained models on TFRD subsets and evaluating their interpolation accuracy (Table VIII). The results show that NIER pre-trained on default Mathit outperforms those pre-trained on the Mathit variants, particularly on the HSink and DSine datasets. The performance on the ADlet dataset was comparable across variants. Overall, these results lend support to the observed trend.

Note that ADlet’s data distribution is simpler than HSink and DSine’s, as all temperature fields in ADlet have a constant temperature (298 K) at the three boundaries, while HSink and DSine have more complex boundary conditions [3]. This suggests that in scenarios with more complex function distributions, the default Mathit with all operators provides a more diverse function distribution for pre-training, leading to stronger generalization and transferability capabilities.

**Ablations on Pre-Training Datasets:** In addition to examining Mathit’s distribution, we evaluated the impact of different pre-training datasets. We pre-trained NIER on each TFRD subset, then fine-tuned and evaluated it on other subsets, with results shown in Table IX. These results unambiguously reveal that NIER pre-trained on Mathit consistently outperforms those pre-trained on other subsets across all three datasets. Given that TFRD subsets relate to the temperature field reconstruction domain and share similar function distributions, they represent stronger baselines compared to Mathit. This underscores the superior generalization capability conferred by our proposed Mathit pre-training dataset.

**VI. DISCUSSION: CONNECTIONS TO TRADITIONAL APPROACH**

The traditional interpolation algorithm constructs the target function as a linear combination of a series of basis functions, usually each of which corresponds to an observed point. For example, to interpolate \( n \) observed points \( O = \{(x_i, y_i)\}_{i=1}^{n} \), the RBF approach uses the following interpolation function

\[
 f_{\text{RBF}}(x) = \sum_{j=1}^{n} \lambda_j \phi(x, x_j).
\]

Here, \( \phi(x, x_j) \) represents a radial basis function specified by the observed point \( x_j \), and \( \lambda_j \) is the coefficient, which can be determined through solving the linear equations \( \sum_{j=1}^{n} \lambda_j \phi(x_i, x_j) = y_i \), \( (x_i, y_i) \in O \) [42].

NIERT can be cast as such an instance of interpolation algorithm using basis functions by drawing the following parallels. The core of the NIER interpolator, i.e., the self-attention with one-sided attention mask shown in (4) can be rewritten as:

\[
 f_{\text{Attn}}(x) = \sum_{j=1}^{n} \alpha(q(x), k_j) v_j,
\]

where \( \alpha(q(x), k_j) \) is the normalized attention weight that models the contribution to query vector \( q(x) \) by the key vector \( k_j \), in which \( k_j \) and \( v_j \) are dominated by the observed point \( x_j \).

Comparing (5) with (6), we can find that self-attention with one-sided attention mask is a general form of RBF interpolation function: \( \lambda_j \) in (5) corresponds to \( v_j \) in (6), and \( \phi(x, x_j) \) in (5) corresponds to \( \alpha(q(x), k_j) \) in (6). This correspondence indicates that \( \alpha(q(x), k_j) \) can be treated as a learnable basis function and, similarly, \( v_j \) can be treated as a predictable basis function coefficient. From this point of view, self-attention with one-sided attention mask is a learnable layer that interpolates the representation of observed points to yield new representations of data points. Together, this insight provides a plausible explanation of our masked self-attention, which is mutually supportive with the interpretability analysis of the contribution of observed points shown in Section V-C.

**VII. CONCLUSION**

We present in the study an accurate approach to numerical interpolation for scattered data. The specific features of our NIER approach are highlighted by the full exploitation of the correlation between observed points and target points through unifying scattered data representation. At the same time, the use of the one-sided attention mask mechanism can effectively avoid the interference of target points onto the observed points. We also propose to use the large-scale synthesis of generic and diverse mathematical functions to build pre-trained NIER interpolators with general interpolation capabilities. The advantages

---

**TABLE VIII**

**THE EFFECT OF MODIFYING THE DISTRIBUTION OF THE PRE-TRAINING FUNCTIONS OF MATHIT ON GENERALIZATION PERFORMANCE ON TFRD**

| Removed operators of Mathit variant | NIER /w pretraining performance on TFRD test set (MAE × 10^{-2}) |
|-------------------------------------|---------------------------------------------------------------|
| [sin, cos, exp, -2, -3]             | HSink 13.943 ADlet 1.780 DSine 4.909 |}

We modified the function distribution of Mathit by excluding certain operators when synthesizing symbolic functions. Then the pre-trained NIERs on these Mathit variants are fine-tuned on TFRD subsets and evaluated to assess the generalization obtained through pre-training.

**TABLE IX**

**THE EFFECT OF DIFFERENT PRE-TRAINED DATASETS ON GENERALIZATION PERFORMANCE ON TFRD SUBSETS**

| Pre-training dataset | NIER /w pretraining performance on TFRD test set (MAE × 10^{-2}) |
|----------------------|---------------------------------------------------------------|
| HSink                | 15.268 3.179 5.894 |
| ADlet                | 15.975 2.760 5.019 |
| DSine                | 13.943 1.780 4.909 |

In addition to the default pre-training dataset Mathit, we pre-train NIER on each subset of TFRD, fine-tune on other subsets, and evaluate interpolation accuracy.
of NIERT in interpolation accuracy have been clearly demonstrated by experimental results on both synthetic and real-world datasets. We expect NIERT, with extensions and modifications, to greatly facilitate numerical interpolations in a wide range of engineering and science fields.

REFERENCES

[1] R. Franke and G. M. Nielson, “Scattered data interpolation and applications: A tutorial and survey,” Geometric Model., pp. 131–160, 1991.

[2] G. Liu, “An overview on meshfree methods: For computational solid mechanics,” Int. J. Comput. Methods, vol. 13, no. 05, 2016, Art. no. 1630001.

[3] X. Chen, Z. Gong, X. Zhao, W. Zhou, and W. Yao, “A machine learning surrogate modeling benchmark for temperature field reconstruction of heat source systems,” Sci. China Inf. Sci., vol. 66, no. 5, pp. 1–20, 2023.

[4] D. Babiri and C. Pecora, Particle Tracking Velocimetry, vol. 785, Bristol, U.K.: IOP Publishing, 2020.

[5] M. Lepot, J.-B. Aubin, and F. H. Clemens, “Interpolation in time series: An introductory overview of existing methods, their performance criteria and uncertainty assessment,” Water, vol. 9, no. 10, 2017, Art. no. 796.

[6] S. N. Shukla and B. M. Marlin, “Interpolation-prediction networks for irregularly sampled time series,” 2019, arXiv: 1909.07782.

[7] M. T. Heath, Scientific Computing: An Introductory Survey, Revised Second Edition. Philadelphia, PA, USA: SIAM, 2018.

[8] R. Bulirsch, J. Stoer, and J. Stoer, Introduction to Numerical Analysis, vol. 3, Berlin, Germany: Springer, 2002.

[9] M. Garnelo et al., “Conditional neural processes,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 1704–1713.

[10] H. Kim et al., “Attentive neural processes,” in Proc. Int. Conf. on Learn. Representations, 2019, pp. 1–18.

[11] J. Lee, Y. Lee, J. Kim, E. Yang, S. J. Hwang, and Y. W. Teh, “Bootstrapping neural processes,” in Proc. Adv. Neural Inf. Process. Syst., H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds, Curran Associates, Inc., 2020, pp. 6606–6615.

[12] B.-J. Lee, S. Hong, and K.-E. Kim, “Residual neural processes,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 253–264, 1978.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, arXiv:1810.04805.

[14] T. Lin, Y. Wang, X. Liu, and X. Qiu, “A survey of transformers,” AI Open, vol. 3, pp. 111–132, 2022.

[15] H. Bao, L. Dong, and F. Wei, “BEiT: BERT pre-training of image transformers,” 2021, arXiv:2106.08254.

[16] C. A. Hall and W. W. Meyer, “Optimal error bounds for cubic spline interpolation,” J. Approximation Theory, vol. 16, no. 2, pp. 105–122, 1976.

[17] W. J. Gordon and J. A. Wixom, “Shepard’s method of "metric interpolation," in Proc. Adv. Neural Inf. Process. Syst., 1987, pp. 5998–6008.

[18] A. Radford et al., “Language models are few-shot learners,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, pp. 1877–1901, 2020.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” 2019, [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

[20] T. Brown et al., “Language models are few-shot learners,” Adv. Neural Inf. Process. Syst., vol. 33, pp. 1877–1901, 2020.

[21] T. Brown et al., “Language models are few-shot learners,” in Proc. Adv. Neural Inf. Process. Syst., H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds, Curran Associates, Inc., 2020, pp. 1877–1901.

[22] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” 2021, arXiv:2111.06377.

[23] L. Biggio, T. Bendinelli, A. Neitz, A. Lucehi, and G. Parascandolo, “Neural symbolic regression that scales,” in Proc. Int. Conf. Mach. Learn., PMLR, 2021, pp. 936–945.

[24] M. Valipour, B. You, M. Panju, and A. Ghodzi, “SymbolicGPT: A generative transformer model for symbolic regression,” 2021, arXiv:2106.14131.

[25] G. Lample and F. Charton, “Deep learning for symbolic mathematics,” 2019, arXiv:1912.01412.

[26] A. Sciacchitano et al., “Collaborative framework for PIV uncertainty quantification: Comparative assessment of methods,” Meas. Sci. Technol., vol. 26, no. 7, 2015, Art. no. 074004.

[27] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, “Predicting in-hospital mortality of ICU patients: The physionet/competing in cardiology challenge 2012,” in Proc. IEEE Comput. Cardiol., 2012, pp. 245–248.

[28] P. Virtanen et al., “SciPy 1.0: Fundamental algorithms for scientific computing in Python,” Nat. Methods, vol. 17, pp. 261–272, 2020.

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 2014, arXiv:1412.3555.

[30] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, “Neural ordinary differential equations,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 6572–6583.

[31] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud, “Latent ordinary differential equations for irregularly-sampled time series,” in Proc. Adv. Neural Inf. Process. Syst., H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds, Curran Associates, Inc., 2019, pp. 5320–5330.

[32] F. Aurenhammer, “Voronoi diagrams—a survey of a fundamental geometric data structure,” ACM Comput. Surv., vol. 23, no. 3, pp. 345–405, 1991.

[33] J. Solomon, Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics. Boca Raton, FL, USA: CRC Press, 2015.

Shizhe Ding received the BE degree from the University of Chinese Academy of Sciences in 2019. He is currently working toward the PhD degree with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China. His research interests lie in data mining and deep learning.

Boyang Xia received the BS degree from the Beijing University of Posts and Telecommunications in 2020. He is currently working toward the master’s degree with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). His research interests include deep learning and data mining.
Milong Ren received the BS degree from Shandong University in 2021. He is currently working toward the PhD degree with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). His research interests include deep learning and bioinformatics.

Dongbo Bu received the PhD degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He is now a professor with the institute and his research interests include AI-aided algorithm design, algorithms for protein structure prediction and protein design. He worked with his colleagues to develop ProFOLD, a software for protein structure prediction, ProDESIGN, a software for protein design, and AIA-ILP, a software for solving integer linear programming with aid of AI.