Journey over destination: dynamic sensor placement enhances generalization

MACHINE LEARNING
Science and Technology

PURPOSE-LED PUBLISHING™ Part of the MACHINE LEARNING Series

PAPER • **OPEN ACCESS**

Journey over destination: dynamic sensor placement enhances generalization

To cite this article: Agnese Marcato *et al* 2024 *Mach. Learn.: Sci. Technol.* **5** 025070

IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 025070 https://doi.org/10.1088/2632-2153/ad4e06

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
19 March 2024

REVISED
17 April 2024

ACCEPTED FOR PUBLICATION
20 May 2024

PUBLISHED
18 June 2024

PAPER

Journey over destination: dynamic sensor placement enhances generalization

Agnese Marcato*, Eric Guiltinan, Hari Viswanathan, Daniel O'Malley, Nicholas Lubbers and Javier E Santos

Los Alamos National Laboratory, Los Alamos, NM, United States of America
* Author to whom any correspondence should be addressed.

E-mail: amarcato@lanl.gov

Keywords: sensors, neural networks, attention, differentiable

Abstract

Reconstructing complex, high-dimensional global fields from limited data points is a challenge across various scientific and industrial domains. This is particularly important for recovering spatio-temporal fields using sensor data from, for example, laboratory-based scientific experiments, weather forecasting, or drone surveys. Given the prohibitive costs of specialized sensors and the inaccessibility of certain regions of the domain, achieving full field coverage is typically not feasible. Therefore, the development of machine learning algorithms trained to reconstruct fields given a limited dataset is of critical importance. In this study, we introduce a general approach that employs moving sensors to enhance data exploitation during the training of an attention based neural network, thereby improving field reconstruction. The training of sensor locations is accomplished using an end-to-end workflow, ensuring differentiability in the interpolation of field values associated to the sensors, and is simple to implement using differentiable programming. Additionally, we have incorporated a correction mechanism to prevent sensors from entering invalid regions within the domain. We evaluated our method using two distinct datasets; the results show that our approach enhances learning, as evidenced by improved test scores.

1. Introduction

The goal of sparse data reconstruction is to take a few sensor values from a space that cannot be fully observed, and use them to reconstruct the global field [1]. Reconstructing a field from a few observations is considered a 'grand challenge' [2–4] in many fields, including: industry [5], medicine [6], and environmental sciences [7–10]. Sparse data reconstruction is especially challenging in applications involving fluid dynamics, which tend to present highly non-linear and chaotic phenomena [11].

Several techniques have been developed to perform this task. These techniques can be based on partial differential equations (PDE) or they can be purely data-driven methods. When attempting to model phenomena governed by PDE, integrating sensor measurements directly into the model can be difficult. In the literature, inverse problems based on finite element methods [12] and on finite difference methods [13] have been developed, but they have only been shown to work on benchmark problems like structural and heat transfer problems. Physics-informed neural networks represent an advanced methodology for integrating sensor data with PDE through automatic differentiation [14, 15]. To effectively apply this method, an accurate understanding of the physical phenomena involved is essential. This requirement ensures that the models can accurately capture the underlying processes and provide reliable predictions. On the other hand, this limits the technique to data where the underlying phenomenon is readily and accurately described by a PDE. Many real-world scenarios (e.g. weather and climate models) are too complex to be readily described by a single PDE.

Purely data-driven sparse sensing has generated a lot of interest since it is a more general approach to tackle this complex issue [2]. Many techniques exist, including simulated annealing [16] and a family of

models that utilize dimensionality reduction to make sparse sensing tractable [17]. This includes reduced order approaches that use sparse time-resolved measurements in fluid flows for full state estimation [18]. Compressed sensing, an influential framework within this domain, exploits the inherent sparsity of signals to reconstruct them from fewer measurements than traditionally required [19]. Systematic random sampling, a key technique within compressed sensing, was introduced by Süzen *et al* [15] to further reduce the number of measurements necessary to accurately reconstruct a signal or an image from under-sampled data.

Machine learning methods have emerged as data-driven general-purpose tools, that can be used for reconstructing fields from sparse sensor measurements. Numerous studies [20–25] have demonstrated the potential of neural networks to efficiently interpret and utilize sparse data in various contexts. Fully connected neural networks, as discussed in Erichson *et al* [20] on shallow architectures, have been successfully trained on benchmark datasets. However, the rigidity of their architecture poses challenges in dealing with dynamic sensor operations. This limitation comes from the fixed structure of fully connected layers, which can restrict the network's ability to adapt to varying input patterns and sensor configurations that are characteristic of dynamic sensing environments. Also convolutional neural networks have been successfully used by Fukami *et al* [21] for field reconstruction from sparse sensing. These networks space definition is more robust but the main throwback is the rigidity of the spatial structure which relies on a Cartesian grid. This constraint can pose challenges in applications requiring more flexible or non-Cartesian spatial arrangements. Graph neural networks have been trained to predict fields from sensors observations by Alet *et al* [22], these networks circumvent the rigid data structure implicit in other models. However, the selection of an appropriate graph topology and the tuning of hyperparameters require careful consideration and customization based on the specific characteristics of the dataset.

State-of-the-art neural networks employ attention mechanisms [26], that have greatly improved over other architecture baselines for a variety of problems [27–29]. These networks employ the attention mechanism to focus on the relevant parts of the input data which enables them to effectively handle long-range dependencies and complex data relationships with improved performance in tasks like language translation and image recognition [30–32]. Recently, attention based neural networks have also demonstrated great performance in field reconstruction tasks [33]. These networks employ positional encoding to accurately map sensor positions, eliminating the need for a Cartesian grid-based data structure or fine-tuned graph structures.

In general, machine learning models use a subset of the dataset as training set, so the sensor positions and field values are known and employed to train the model [20, 21, 34, 35]. Despite variations in their architectures, a shared characteristic in these models is the static positioning of sensors throughout the training phase. In fact, the current literature still lacks a comprehensive and end-to-end strategy that exploits sensor positioning during training to improve field reconstruction performance.

This study introduces a methodology that allows a model to reconstruct a field by adjusting sensors position through backpropagation, enhancing the model's capability to efficiently explore the spatial domain. The sensors location are trained alongside the parameters of the attention-based neural network: to perform this we developed a fully end-to-end differentiable framework, as shown in the workflow of figure 1(A). We call this strategy *differentiable walk*, which outperformed two other strategies we tested in this study: the *global random sampling* and *random walk* strategy, as illustrated in figure 1(B) and detailed in the next sections. These novel strategies are presented in this paper for the first time. In the differentiable walk strategy the optimizer used to train the neural network is employed to adjust the sensors location alongside the network parameters. This approach enables efficient exploitation of the training dataset by allowing the network to move sensor positions during training. By enabling the neural network to optimize the locations autonomously, it gains an enhanced spatial awareness leading to improved field reconstruction performance. Moreover, we corrected our algorithm to prevent sensor movement into invalid domain areas. In this work, we implemented these strategies to an attention-based neural network, however, they are versatile and can be adapted for use with any type of neural network. Our study reveals a significant enhancement in generalization ability over existing benchmarks. The differentiable walk strategy shows state-of-the-art performance in two datasets: vorticity reconstruction for a Kármán vortex street fluid dynamic case, and global sea surface temperature reconstruction. Nonetheless, this methodology is data-agnostic, as it was not developed for specific datasets.

2. Neural network architecture

While the idea of exploiting the training set efficiently through sensors change of location during training is general and could be implemented along with any neural network architecture, we employed an attention based neural network [26, 36]. As mentioned, the attention mechanism can be useful because of its ability to model the relationships between any pair of input values, weighting dynamically the significance of the input

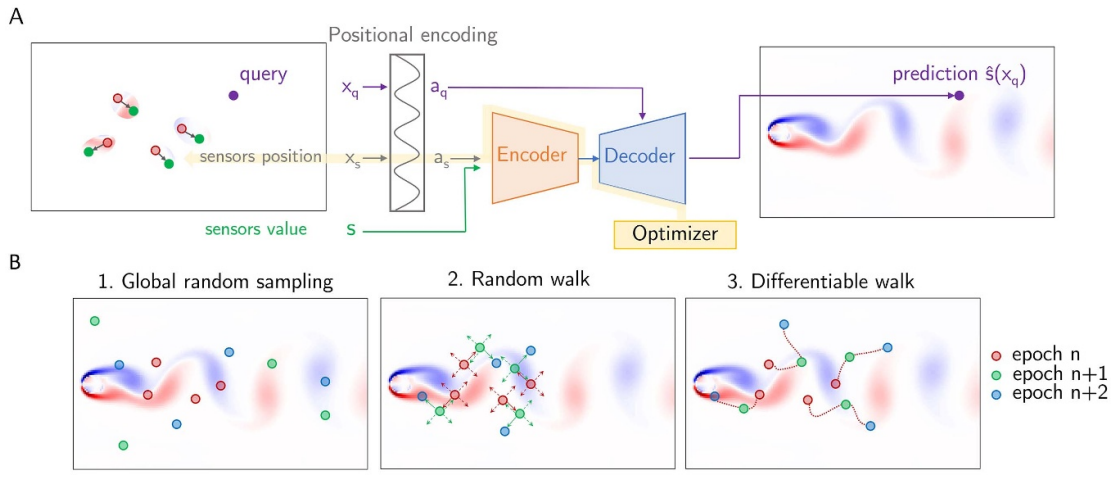

Figure 1. (A) A schematic representation of the proposed end-to-end differentiable framework for sensor placement improvement. The model input consists of the locations of sensors, which are represented through positional encodings. This encoded information is then compressed into a latent space using an encoder. The decoder then queries this latent space for new positions to generate their field values. (B) Strategies adopted in this study for dynamic positioning of the sensors during the training of the neural network: (1) *Global random sampling*—randomly selects new sensor positions across the domain at the start of each optimization step; (2) *Random walk*—moves sensors by a small distance in a random direction; (3) *Differentiable walk*—treats sensor positions as adjustable network parameters, optimized through backpropagation during training.

features by computing attention scores [27, 29, 37]. While traditional attention mechanisms face computational inefficiencies due to their quadratic complexity with input size, our approach incorporates the Senseiver architecture [33], an adaptation of the Perceiver IO framework [36] to avoid this issue. Unlike the conventional attention models, the Perceiver IO, and the Senseiver, utilize cross-attention with latent arrays to maintain computational efficiency, irrespective of input scale.

The Senseiver rather than treating field reconstruction as a dense image problem, encodes the sparse observed sensor values into a fixed sized vector, which can then be used with a decoder to estimate the entire field. Crucially, it is agnostic to spatial dimensionality. The model can be trained without specialized feature engineering [38], i.e. no problem-specific processing of input features or labels is required, since the direct sensor observations can be used as input to and output from the network. Furthermore, it has been shown that attention based neural networks can deliver high-quality reconstruction results with fewer parameters than other network types. This performance results in a trained model that is less computationally intensive, offering a more resource-effective solution in data processing and analysis.

The Senseiver is trained to learn compact representations of system states from a small number of sensor observations at a given time, that is the input of the model. Then, the encoded representation is used to decode the state of the full system for new queries. The attention based neural network model, as depicted in the inner part of figure 2, comprises three primary elements:

(i) A positional encoder, P_{E}, which translates a spatial coordinate $\mathbf{x}_{\mathrm{s}}$ into a set of sine-cosine spatial encodings [26], $\mathbf{a}_{\mathrm{s}}$, facilitating the conversion of an n-dimensional spatial location into a vector:

$$\mathbf{a}_{\mathrm{s}} = P_{\mathrm{E}}\left(\mathbf{x}_{\mathrm{s}}\right). \tag{1}$$

(ii) An attention-driven encoder, $E(\mathbf{s}, \mathbf{a}_{\mathrm{s}})$, which processes the spatial encodings of sensor locations $\mathbf{a}_{\mathrm{s}}$ and their corresponding values $\mathbf{s}$, yielding a latent matrix $\mathbf{Z}$. This matrix serves as a condensed representation of the system at a specific time, t:

$$\mathbf{Z} = E\left(\mathbf{s}, \mathbf{a}_{\mathrm{s}}\right). \tag{2}$$

(iii) An attention-focused decoder, $D(\mathbf{Z}, \mathbf{a}_q)$, that predicts the field value at any given query position $\mathbf{a}_q$, whose position is also translated into a set of spatial encodings:

$$\hat{\mathbf{s}}\left(\mathbf{x}_q, t\right) = D\left(\mathbf{Z}, \mathbf{a}_q\right) = D\left(\mathbf{Z}, P_{\mathrm{E}}\left(\mathbf{x}_q\right)\right). \tag{3}$$

The training set is comprised of sensors and field values. The predicted values associated with the queries locations are compared to the actual values to compute the loss. The architecture details and the main hyperparameters are listed in table 1.

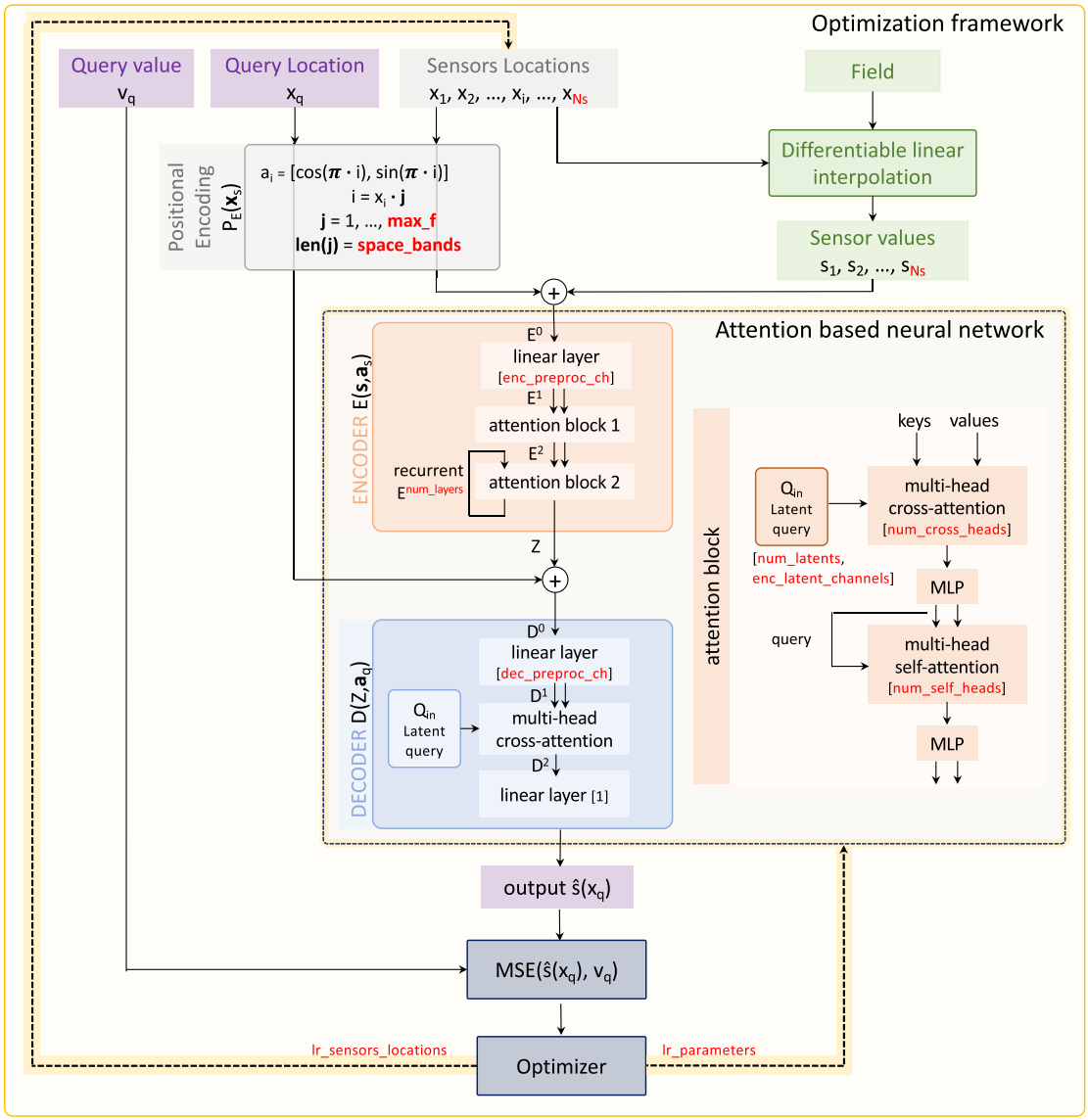

Figure 2. Optimization framework. The sensor positions in the training set are a trainable parameter for the optimizer, during each optimization step the parameters of the attention based neural network and the sensor positions are updated through this differentiable workflow. In red the hyperparameters—the values employed in this work are summarized in table 1.

3. Moving sensor strategies

Exploiting the training dataset efficiently is important to develop models with robust generalization capabilities on new inputs. When the sensors are allowed to move during the training phase, the encoder can better explore the spatial domain, enhancing its learning efficacy. The three different strategies we tested are the following.

Global random sampling. A straightforward method to enable sensor movement during training is through a global random sampling strategy, where a new set of sensor positions is picked across the domain at the beginning of each optimization step (as illustrated in figure 1(B.1)). In this strategy, the relocation of sensors is entirely independent across different steps, so there is no pattern guiding the sensor movements from one epoch to the next.

Random walk. Our second strategy involves a controlled randomness to sensor movement: during each optimization step, each sensor can move in a random direction but the distance from the starting position is fixed (as illustrated in figure 1(B.2)). In this way the sensor's trajectory through the space is less unpredictable, and the movement in the space can be exploited to capture and learn from the spatial relationships in the field in a more meaningful way compared to completely random positions.

Table 1. Architecture details and training hyperparameters employed in the optimization workflow of figure 2.

Hyperparameter	Value
Number of sensors for the cylinder dataset [Ns]	4-8-16
Number of sensors for the sea temperature dataset [Ns]	10-20-30-50-100
Maximum frequency [max_f]	100
Number of frequencies [space_bands]	32
Number of pre-processing channels in the encoder [enc_preproc_ch]	16
Number of recurrent layers [num_layers]	3
Number of cross attention heads [num_cross_head]	16
Number of latents [num_latents]	256
Number of latent channels [latent_channels]	16
Number of self attention heads [num_self_head]	3
Number of pre-processing channels in the decoder [dec_preproc_ch]	256
Learning rate [lr_parameters]	1×10^{-4}
Learning rate for the sensor positions [lr_sensors_locations]	0.25

Differentiable walk. Our third strategy eliminates randomness entirely by integrating sensor positions as trainable parameters alongside the neural network's weights. This integration allows the optimization algorithm, Adam [39] in this work, to refine both the network's parameters and the sensor locations during the training process. However, it is important to note that other optimizers commonly used for training neural networks can also be incorporated into this workflow with appropriate adjustments. In our early experiments we tested both Adam and stochastic gradient descent without observing appreciable differences in the reconstruction task.

We compared these strategies based on the accuracy in the prediction of the field values at given query positions. To assess the network's performance in achieving this objective, we utilize the mean squared error (MSE) as loss function, comparing the predicted field value at the query positions, $\hat{\mathbf{s}}(\mathbf{x}_q)$, with the actual field values, $\mathbf{v}_q$. Based on the calculated loss at each step of optimization, the optimizer adjusts the network's parameters and, in the case of the differentiable walk strategy, the positions of the sensors.

3.1. Differentiable sensor placement

In the random walk and differentiable walk strategies, sensor positions are updated within the confines of the dataset's domain at the end of each optimization step. This approach allows sensors to move to intermediate positions within the data points available in the training set. Consequently, it is necessary to recalculate both the positional encodings and the values of the property at these updated positions at each optimization step. This recalculation is integral to the continuation of the training process. To ensure seamless integration into the training workflow, these recalculations must be performed in a differentiable manner. A naive approach to get the property values for the new sensor positions would be to use the indexed discrete positions as trainable parameters, then round them to the nearest integer and utilize these to index the field. However, this process, referred to as array indexing, is non-differentiable.

To overcome this limitation, and allow gradient flow, we implemented a differentiable interpolation function to determine the sensors values at precise locations. In particular, given the updated positions of the sensors at each step, the interpolation scheme is employed to compute the values of the property at those points, based on their neighboring data. These calculated sensor values then serve as inputs to our neural network. The interpolation function we implemented and its use in the optimization loop is detailed in algorithm 1. For simplicity in the algorithm we refer to a 2D case, as the datasets we tested for our workflow. Nevertheless the implementation for a 3D dataset is straightforward as the algorithm remains consistent across dimensions.

In figure 2 this optimization workflow is detailed for the differentiable walk case, the hyperparameters in red are detailed in table 1. It is worth noting that the we set a learning rate for the network's parameters and a different one for the sensor positions, in fact, the latter depends on the order of magnitude of the predicted output. We employed the same learning rates for both our datasets since we normalized the field values to span between 0 and 1.

In all our runs, training stops once the training loss plateaus, indicating that no further improvement in the model's accuracy is expected.

Algorithm 1. Differential interpolation with spatial encoding and optimization.

Input: $x_{sens}, y_{sens}, field, x_{init}, y_{init}, lr, num_{epochs}, NeuralNetwork, SpatialEncoding$
Output: Optimized model
Initialize x_{sens}, y_{sens} with x_{init}, y_{init}, enable gradient computation
Initialize Model ← NeuralNetwork()
Initialize Optimizer (e.g. Adam) with parameters and lr
function DIFFINTERPOLATION ($x_{sens}, y_{sens}, field$)
    $X, Y \leftarrow$ range(0, size of $field$)
    $size_x, size_y \leftarrow field.shape$
    $index_x \leftarrow floor(x_{sens} * size_x)$
    $index_y \leftarrow floor(y_{sens} * size_y)$
    $w_{sum} \leftarrow 0$
    for (i, j) in $[(0,0),(0,1),(1,0),(1,1)]$ **do**
        $w \leftarrow (X[index_x + i] - x_{sens}) \times (Y[index_y + j] - y_{sens})$
        $w \leftarrow -w$ if $i \neq j$ else w
        $w_{sum} \leftarrow w_{sum} + w \times field[index_x + (1-i), index_y + (1-j)]$
    end for
    return w_{sum}
end function
$x_{sens}, y_{sens} \leftarrow$ RandomLocation()
for $epoch \leftarrow 1$ to num_{epochs} **do**
    $x_{query}, y_{query} \leftarrow$ RandomLocation()
    $f_{sens} \leftarrow$ DiffInterpolation($x_{sens}, y_{sens}, field$)
    $e_{sens} \leftarrow$ SpatialEncoding(x_{sens}, y_{sens})
    $e_{query} \leftarrow$ SpatialEncoding(x_{query}, y_{query})
    $f_{pred} \leftarrow$ Model($e_{sens}, f_{sens}, e_{query}$)
    Calculate MSE loss and perform backpropagation
    Update model and sensor coordinates with optimizer
end for

4. Flow past a cylinder dataset

The first dataset we analyzed was derived from a simulation of two-dimensional unsteady flow navigating a cylindrical obstacle [40]. The resulting simulation manifests a phenomenon known asvon Kármán vortex street -an alternating pattern of left- and right-handed vortices in the flow field trailing a cylinder. The simulation, which solves the incompressible Navier–Stokes equation at a Reynolds number of 100, was conducted on a computational grid of 192 × 112 with 5000 time frames, which roughly corresponds to four periods of vortex shedding. The simulation domain extends 10 cylinder diameters vertically and 15.7 cylinder diameters horizontally. The focus of data collection was the fluid's vorticity field, influenced by the presence of a solid obstruction.

During the training phase, we randomly selected 50 frames, amounting to 1% of the total data. The performance of the trained models was assessed using the remaining 4950 frames using as metric the L_2 norm, which is defined as follows using the nomenclature of figure 2:

$$L_2\,\text{norm} = \frac{\|\mathbf{v}_q - \hat{\mathbf{s}}\left(\mathbf{x}_q\right)\|_2}{\|\mathbf{v}_q\|_2}. \tag{4}$$

4.1. Strategies comparison

Initially, we evaluated the performance of the strategies outlined in the previous section, namely global random sampling, random walk, and differentiable walk. Each strategy incorporates movement of sensor positions during neural network training, and their outcomes were compared with those obtained from training with fixed sensor positions, as documented in the original Senseiver study [33]. This comparative analysis allows us to understand the impact of sensor mobility on the neural network's ability to reconstruct the field accurately. In figure 3, the fixed position strategy and the moving sensors strategies have been compared based on the L_2 norm error distribution of the test set frames. The comparison incorporates both the mean (indicated by a dashed line) and the maximum (indicated by a solid line) values of the error distribution as key metrics. The trainings have been performed using four sensors, whose initial position is the one displayed in figure 4(B).

Using the fixed position strategy (in blue) as a baseline for comparison, it is observed that the global random sampling strategy (represented in grey) underperforms, with its error distribution being shifted an order of magnitude higher. In this context, the movement of sensors is entirely random and uncorrelated

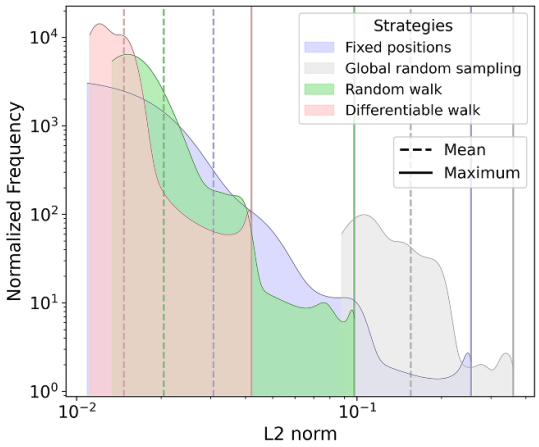

Figure 3. The comparison of training strategies is conducted through density plots illustrating the L2 norm of the discrepancy between actual and predicted vorticity in frames of the flow past a cylinder dataset, with the training utilizing four sensors. The density plots represent kernel density estimates of the distribution, the dotted lines denote the average values of the distribution, while the solid lines highlight the maximum error within the distribution. When the distribution is shifted towards lower errors (left of the plot), the accuracy of the predictions is increased.

across epochs, introducing a level of unpredictability that may not be useful to optimal learning. Such complete randomness in sensor positioning can lead to inefficiencies in training, as the model may struggle to extract meaningful spatial relationships or patterns from the data. This approach, while ensuring varied exposure, might influence negatively the model's ability to develop a coherent understanding of the spatial domain, compromising its generalization capabilities on unseen inputs.

As opposed to the global random sampling approach, the random walk strategy (in green) demonstrates an enhancement over the fixed position baseline. This is evidenced by a tighter error distribution, leading to consistently lower mean and maximum errors. Unlike the unconstrained randomness in sensor movement seen with global random sampling, the randomness in the random walk strategy is restricted to the direction of movement. This constraint facilitates a more systematic exploration of the spatial domain by the attention based neural network, resulting in a more effective utilization of the training set data and, consequently, improved model performance.

The third strategy we investigate, named differentiable walk (in red), eliminates sensor movement randomness, instead employing a method where sensor positions are optimally adjusted by the network's optimization algorithm. This approach yields the best outcomes regarding error distribution on the test set, with the L2 norm errors shifting towards lower values (left of the plot), resulting in both mean and maximum errors being the lowest observed in our study.

The improved performance of the differentiable walk strategy can be attributed to the network taking an active role in leading the exploration of the spatial domain. Compared to strategies that rely on predefined or random sensor movements, differentiable walk allows the optimization algorithm to intelligently adjust sensor positions. This guided exploration ensures that the network focuses on areas of the spatial domain that are most informative for improving its predictions by leveraging the network's ability to discern and prioritize these areas. This approach highlights how effectively using the network to guide sensor placement can greatly improve the accuracy of field reconstructions.

Given its best performance, we have identified the differentiable walk strategy as the most effective method for adjusting sensor positions during neural network training. Consequently, we have chosen to focus on this strategy for detailed analysis and discussion in subsequent results.

4.2. Differentiable walk strategy results

We demonstrated that the differentiable walk strategy outperforms other methods, making it the optimal approach for moving sensors during the training. To support this, we conducted an in-depth comparison with the fixed positions strategy using the flow past cylinder dataset. Our objective is to evaluate the performance of our methodology across various scenarios, specifically focusing on the impact of the number of sensors employed and their initial placement. This analysis is crucial for understanding how our strategy adapts to different configurations, ensuring its effectiveness and reliability in diverse conditions.

We trained the system using 4, 8, and 16 sensors, and we evaluated their initialization across four distinct configurations: literature locations [21] (strategic points where the field presents high variation), figure 4(B) and three random locations within the domain. The results are summarized in the plot of figure 4(A). In this

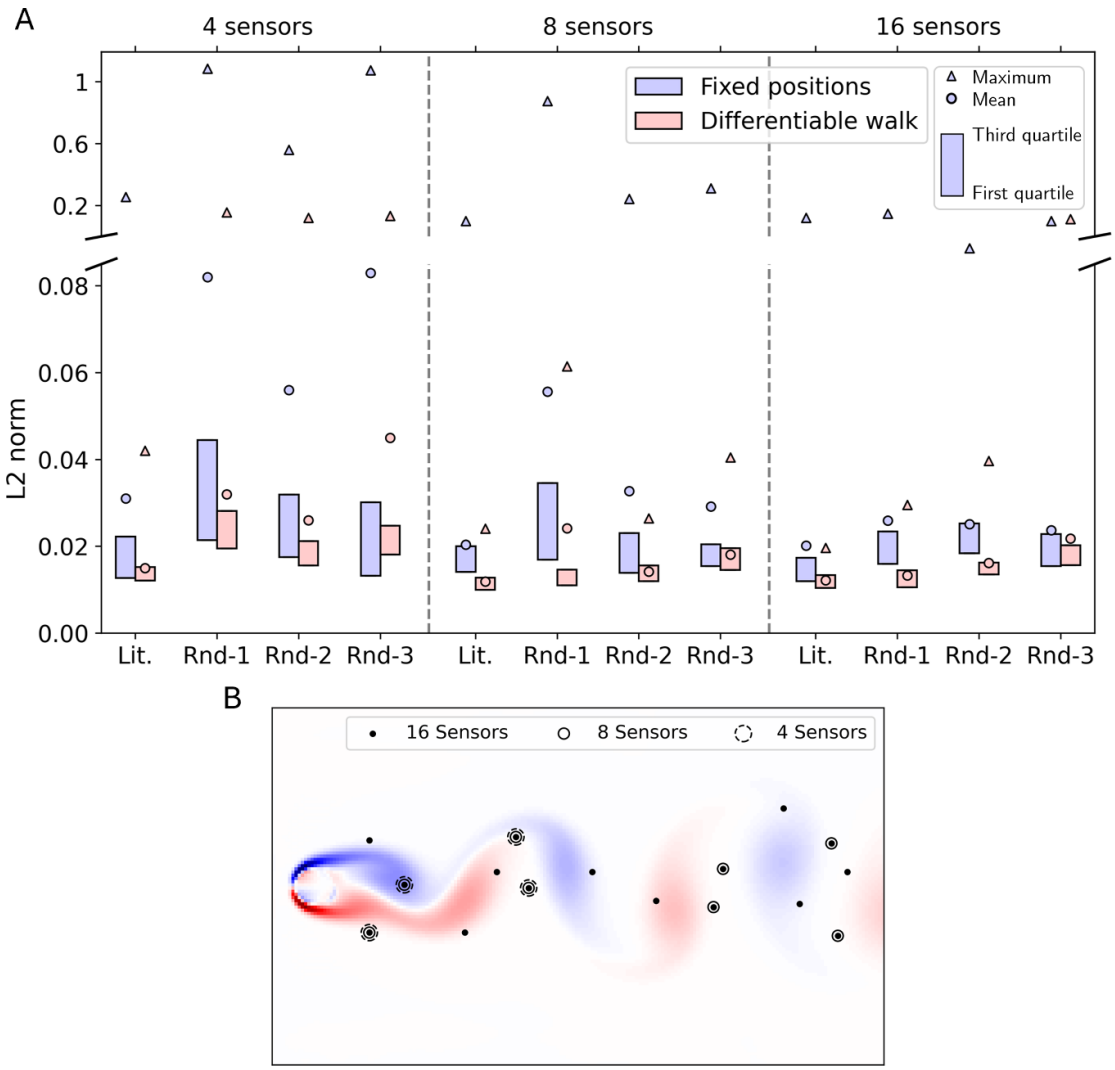

Figure 4. (A) Boxplot diagram of the L2 norm for the flow past cylinder dataset. Results for 4, 8 and 16 sensors, for each set of sensors a different initial position of the sensors is employed as starting point for the training: literature positions, and three sets of random positions. (B) Position of the sensors from the literature (Lit.) [21].

plot the distribution of the L_2 norm in the test frames of the dataset is visualized in terms of first and third quartiles, mean and maximum of the L_2 norm distribution, as detailed in the top right legend. The fixed position (in blue) and the differentiable walk (in red) strategies are compared. It's crucial to highlight that at inference for the differentiable walk strategy we employed as sensor locations the one corresponding to the optimal loss recorded during training. This means that the sensors were placed differently from their initial positions, unlike the fixed position strategy models where inference is based on the sensors' original locations.

When the number of sensors is increased from four to eight, the reconstruction accuracy improves: the mean and maximum errors decrease, so the distribution of the L2 norm error is narrower. The further increase of sensors from eight to sixteen yields a general improvement in accuracy, though with less consistency. It's important to note that our evaluation is based on a single training instance for each case, thus some variance is anticipated. Also, when the sensor positions is initialized in strategic location in the domain (literature positions), the error decreases leading to the best performance. These trends are coherent for both strategies, but the differentiable walk strategy highly improves the field reconstruction capability of the network: as seen in the comparison of the percentiles, as well as the mean and the maximum of the L_2 norm distributions.

4.3. State-of-the-art comparison

The inference results discussed so far were generated using as sensor locations the one adjusted at the epoch corresponding to the lowest loss observed. For a thoroughly equitable comparison with existing literature accuracy results, we inferred our model using the same sensor positions, so the initialized locations. To

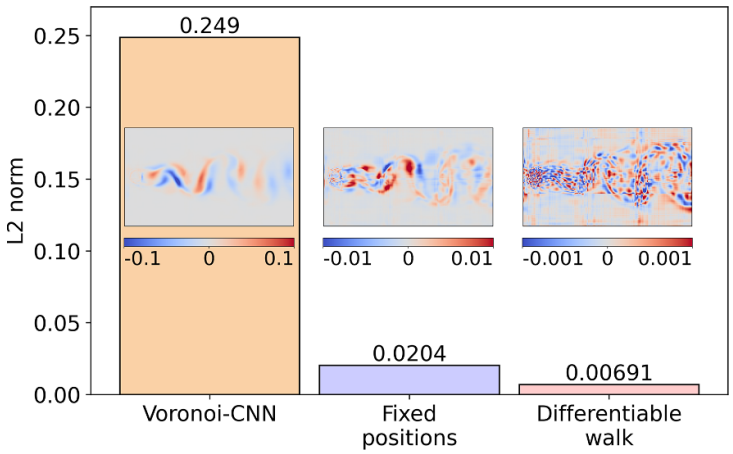

Figure 5. Test error comparison between Voronoi-CNN [21], the fixed positions and differentiable walk strategies. The test set is made by 4950 frames (95% of the total dataset) and eight sensors are used to predict the field, the sensor positions used for the inference is the one of figure 4(B). Snapshots of pixel wise errors are incorporated for the three strategies, note that the order of magnitude of the scales is different for the three snapshots. The Voronoi-CNN serves as a baseline, while the Fixed positions method achieves a 91.8% improvement over the baseline. The differentiable walk startegy further enhances the performance by 66.1%.

achieve this, we further refined our approach by training the differentiable walk strategy model in reverse. This process aimed to minimize the loss associated with the initial sensor positions.

Subsequent inferences made with these original positions confirmed, as illustrated in the figure 5, a significant enhancement in performance of the differentiable walk strategy. In the plot, three strategies are compared: the Voronoi-tessellation-assisted convolutional neural network by Fukami *et al* [21], the Senseiver by Santos *et al* [33] which achieves a 91.8% improvement, and the approach of this work that further enhances the Senseiver's performance by 66.1%. The case study was the flow past cylinder case and the same set of eight sensors is employed to evaluate all models under the same testing conditions.

5. Sea temperature forecasting

The second dataset that we used to test our approach is the NOAA sea surface temperature [41]. This real-world dataset was collected from satellite, buoys, and ship-based observations over time. The data comprise weekly observations of the sea surface temperature of the Earth at a spatial resolution of 360×180 (longitude and latitude, respectively) - giving a resolution of one degree longitude and one degree latitude. For training, we use 1040 snapshots spanning from the year 1981 to 2001, and then we tested on snapshots from 2001 to 2018.

The results detailed in this section were obtained by training the attention based neural network through the implementation of the differentiable walk strategy.

5.1. Flipping gradients to avoid invalid regions

The dataset is made by frames of the sea temperature observations, each frame consists of an array in which the sea region is represented by sea temperature values in Celsius degrees, while the land region is denoted by zeros. Clearly, the land region represents an invalid portion of the domain since sensors register a zero value during their optimization process when they end up over land. This presents a challenge for the network as it learns to interpret a temperature of zero Celsius degrees in these regions. This is visualized in figure 6(A). In the plot the 'trajectories' of 10 sensors during training have been plotted, the green dots being their initialization position, and the red dots the final positions at the end of the training. These trajectories show that sensors may undergo several optimization steps within the land region before either exiting or becoming 'trapped' in it. It is worth noting that the movement patterns of the sensors are not governed by physical principles, making their trajectories challenging to decipher. The sensors locations are tuned in conjunction with the network's trainable parameters, adding a layer of complexity to understanding their specific motion behavior.

To avoid the issue of sensors entering the land region, we introduced a gradient flipping correction within the optimizer. Specifically, this entails checking for each sensor their proximity to the land to determine whether it is within a proximity of less than 2 pixels from the land. If this criterion is met, we reverse the direction of the gradient and also adjust the sign of the exponential moving average component

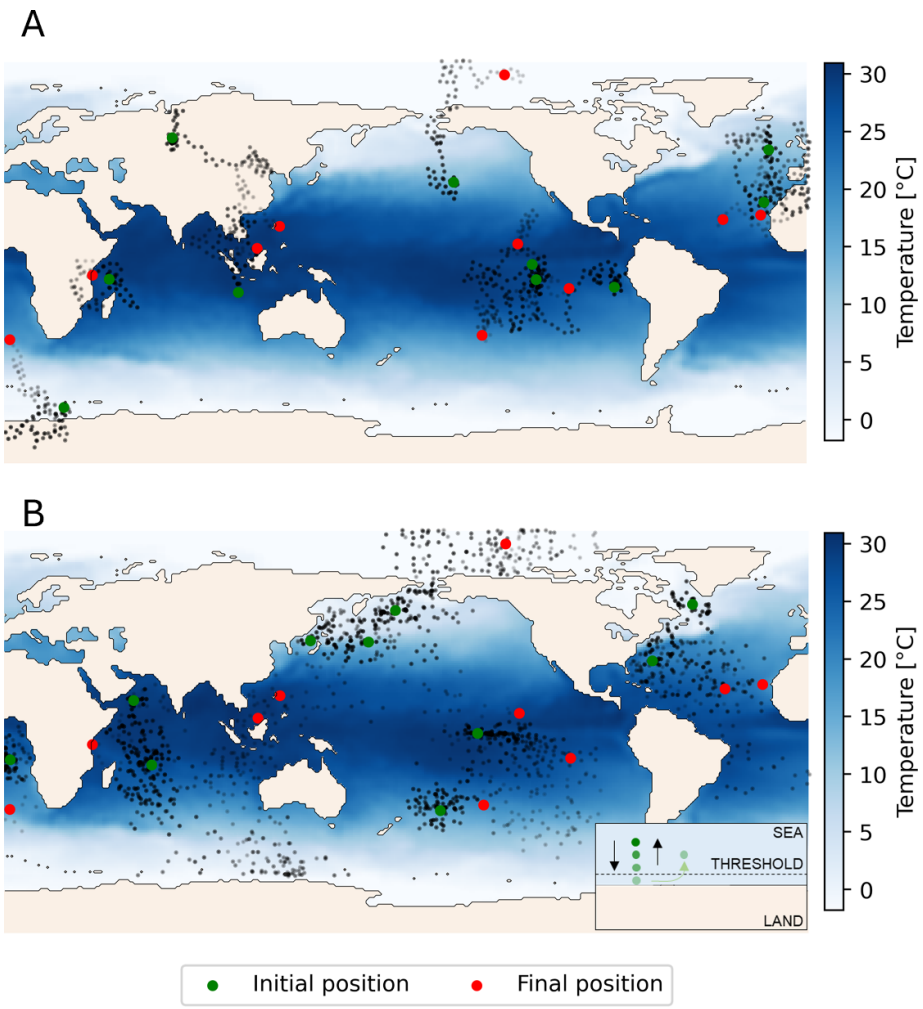

Figure 6. Initial sensor positions are indicated by red markers, with their post-training final positions denoted in green. The paths traveled by the sensors during training are represented by the trails of black dots. (A) It's noteworthy that, due to the momentum component in the Adam optimizer, sensors can travel inland. (B) To avoid forbidden regions in the domain we implemented a gradient flipping. The performance of the trained model in the reconstruction task is quantified in the plot of figure 7 (10 sensors).

of the Adam optimizer. This adjustment is fundamental because the exponential moving average reflects the influence of past gradients in its calculations, thereby flipping it ensures the redirection away from the land region. After reverting the gradient sign, if in the next optimization step the sensor's distance is greater than 2 pixels, it is deemed acceptable. However, if the distance remains less than 2 pixels, we further evaluate the sensor's movement direction: if it is moving away from land, we continue monitoring its distance in subsequent epochs; if moving towards land, we reverse the gradient again and continue monitoring in subsequent epochs. This corrective mechanism is sketched in the bottom right of figure 6(B). As illustrated in the figure, thanks to this intervention, the sensors trajectories no longer enter into the land region. This correction was applied to all trainings using this dataset and their results are detailed in the next paragraphs.

5.2. Differentiable walk results

To evaluate the efficacy of the fixed positions strategy versus the differentiable walk strategy in forecasting sea surface temperature, we analyzed the performance with a progressively increasing number of sensors. Both the moving and fixed training strategies were examined using randomly initialized sensor locations. The results are reported in the violin plots of figure 7 displaying the L_2 norm error distributions of the test set frames for the fixed positions strategy (in blue) and the differentiable walk strategy (in red). The first, second, and third quartiles are reported as horizontal lines in each distribution.

For both the fixed positions and differentiable walk strategies, we tested the performance using 10, 20, 30, 50, and 100 sensors, all initialized randomly with a consistent seed to ensure comparability between strategies. The dynamic optimization of sensor positions during training led to a decrease in test error, as evidenced by the reduction of the error quartiles. The differentiable walk strategy consistently outperformed the fixed positions approach, with the disparity in accuracy becoming particularly pronounced with fewer

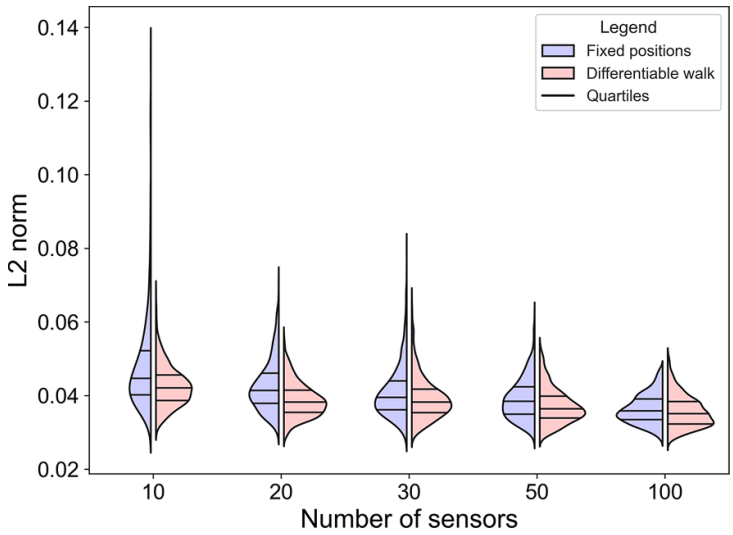

Figure 7. Distribution of the L2 norm error between the actual and the predicted temperature for the frames of the sea temperature dataset. Results for increasing number of sensors (10, 20, 30, 50, 100). For each set the error distribution of the fixed positions and the differentiable walk strategies are compared. The lines in the distributions are the first, second (median), and third quartiles.

sensors. This marked decrease in error rates highlights that our dynamic optimization method significantly reinforces the model's generalization capabilities. For instance, the error distribution's tail for the 10 sensor configuration is significantly narrower with the differentiable walk strategy, indicating a robust improvement in model performance. As the sensor count increases, the performance gap between the two strategies diminishes, in fact, when using 100 sensors the difference becomes less noticeable. While the differentiable walk strategy remains superior even with a high number of sensors, the necessity to move sensors diminishes as the domain becomes adequately represented without adjustments.

6. Discussion and conclusions

In this study, we explored algorithms for dynamically moving sensors during the training of a neural network to enhance training data utilisation and, consequently, model performance. Through comparative analysis of various strategies, we proved that the most effective approach for doing this is through the differentiable walk strategy, that considers the sensor positions as trainable parameters so these are adjusted during the training of the neural network. Moreover our methodology includes the implementation of a gradient flipping correction, designed to keep sensors away from invalid regions of a domain or locations where sensors cannot be placed, thereby ensuring the reliability of data captured for model training -this correction is necessary in a number of applications such earth observations, porous media systems, or industrial devices. This correction mechanism contributes to the robustness of our approach. We demonstrate the versatility and superior performance of the differentiable walk strategy on two distinct benchmark datasets—the flow past a cylinder and sea surface temperature dataset. We compared our inference results to the results from state-of-the-art neural network models, ensuring a fair comparison by using the same training set and the same sensors position for all models during the testing phase. This approach proofs that the differentiable walk, while enhancing the model's learning and predictive accuracy during training, does not necessitate physical sensor movement for data acquisition in practice. Therefore, the enhanced performance achieved with our strategy is due to improved training dynamics rather than the physical repositioning of sensors.

Despite the success of this framework on the examples shown, potential for further research exists. In our framework we use linear interpolation, which may not perfectly capture the complexities of certain fields. This simplification, while beneficial for computational efficiency and ease of implementation, might introduce inaccuracies when dealing with high-frequency variations in the field. In complex or dynamic environments, like turbulent systems, interactions among multiple sensors may have significant effects on the overall system performance, which our current model may not fully account for. Future work could explore extending the algorithm to account for a wider window (multiple neighbors) which could allow better understanding of the local regions, hence allowing the sensors to move more optimally and potentially improving the reconstruction accuracy for more complex problems. Finally, our study utilized sine-cosine encoding for sensor positions, which is not the only potential method for embedding space. Future research

could explore the utility of other position encodings, such as wavelet encodings, to determine if these can offer improved performance. The exploration of alternative encoding methods may yield diverse insights, and could potentially enhance the robustness and versatility of the framework. Hence, we regard this as an important avenue for future exploration.

These limitations notwithstanding, our end-to-end differentiable framework has achieved state-of-the-art results on two benchmark problems. This strategy not only facilitates the strategic motion of sensors for improved field reconstruction but also showcases remarkable adaptability across different applications. Allowing the attention-based neural network to self-adjust sensor positions proves enhances spatial awareness and understanding the field dynamics. It is important to note that we cannot claim to achieve the global optimum for sensor locations. In fact, despite training the model multiple times, we observed consistent global accuracy while the final sensor positions varied. This suggests that the training process of iteratively adjusting sensor positions (*journey*) may be the primary factor driving performance improvements, rather than the specific final positions achieved (*destination*).

Data and code availability statement

No new data were created or analysed in this study.

The code is available at https://github.com/OrchardLANL/Senseiver.

Acknowledgments

JES and AM gratefully acknowledge the support of the Center for Non-Linear Studies (CNLS) for this work. We are grateful to the developers of the many software packages used throughout this project including, but not limited, to PyTorch [42], Numpy [43], and Matplotlib [44].

ORCID iDs

Agnese Marcato https://orcid.org/0000-0002-9934-9004
Javier E Santos https://orcid.org/0000-0002-2404-3975

References

[1] Shen H, Li X, Cheng Q, Zeng C, Yang G, Li H and Zhang L 2015 *IEEE Geosci. Remote Sens. Mag.* **3** 61–85
[2] Manohar K, Brunton B W, Kutz J N and Brunton S L 2018 *IEEE Control Syst. Mag.* **38** 63–86
[3] Jiang G, Kang M, Cai Z, Wang H, Liu Y and Wang W 2022 *Int. J. Thermal Sci.* **175** 107489
[4] Wang B, Deng X, Liu W, Yang L T and Chao H-C 2013 *IEEE Wireless Commun.* **20** 74–81
[5] Fortuna L *et al* 2007 *Soft Sensors for Monitoring and Control of Industrial Processes* vol 22 (Springer)
[6] Paoli A, Neri P, Razionale A V, Tamburrino F and Barone S 2020 *Sensors* **20** 6584
[7] Tian G Y, Sophian A, Taylor D and Rudlin J 2005 *IEEE Sens. J.* **5** 90–96
[8] Rouet-Leduc B, Hulbert C and Johnson P A 2019 *Nat. Geosci.* **12** 75–79
[9] Su H, Jiang J, Wang A, Zhuang W and Yan X-H 2022 *Remote Sens.* **14** 3198
[10] Saint-Vincent P M, Sams I I I J I, Hammack R W, Veloski G A and Pekney N J 2020 *Environ. Sci. Technol.* **54** 8300–9
[11] Mikhaylov K, Rigopoulos S and Papadakis G 2023 *Chem. Eng. Sci.* **279** 118881
[12] Gherlone M, Cerracchio P, Mattone M, Di Sciuva M and Tessler A 2012 *Int. J. Solids Struct.* **49** 3100–12
[13] Gu Y, Wang L, Chen W, Zhang C and He X 2017 *Int. J. Heat Mass Transfer* **108** 721–9
[14] Cai S, Mao Z, Wang Z, Yin M and Karniadakis G E 2021 *Acta Mech. Sin.* **37** 1727–38
[15] Süzen M, Giannoula A and Durduran T 2010 *Opt. Express* **18** 23676–90
[16] Das R 2012 *Int. J. Comput. Fluid Dyn.* **26** 499–513
[17] Zhou H, Soh Y C, Jiang C and Wu X 2015 Compressed representation learning for fluid field reconstruction from sparse sensor observations *2015 Int. Joint Conf. on Neural Networks (IJCNN)* (IEEE) pp 1–6
[18] Loiseau J-C, Noack B R and Brunton S L 2018 *J. Fluid Mech.* **844** 459–90
[19] Donoho D L 2006 *IEEE Trans. Inf. Theory* **52** 1289–306
[20] Erichson N B, Mathelin L, Yao Z, Brunton S L, Mahoney M W and Kutz J N 2020 *Proc. R. Soc.* A **476** 20200097
[21] Fukami K, Maulik R, Ramachandra N, Fukagata K and Taira K 2021 *Nat. Mac. Intell.* **3** 945–51
[22] Alet F, Jeewajee A K, Villalonga M B, Rodriguez A, Lozano-Perez T and Kaelbling L 2019 Graph element networks: adaptive, structured computation and memory *Int. Conf. on Machine Learning* (PMLR) pp 212–22
[23] Abedin A, Rezatofighi S H, Shi Q and Ranasinghe D C 2019 arXiv:1906.02399
[24] Wang G, Jia Q-S, Zhou M, Bi J, Qiao J and Abusorrah A 2022 *Artif. Intell. Rev.* **55** 565–87
[25] Duthé G, Abdallah I, Barber S and Chatzi E 2023 arXiv:2301.03228
[26] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Ł K and Polosukhin I 2017 *Advances in Neural Information Processing Systems* vol 30
[27] Yu J, Wang Z, Vasudevan V, Yeung L, Seyedhosseini M and Wu Y 2022 arXiv:2205.01917
[28] Chowdhery A *et al* 2023 *J. Mach. Learn. Res.* **24** 1–113 (available at: https://jmlr.org/papers/volume24/22-1144/22-1144.pdf)
[29] Jumper J and Hassabis D 2022 *Nat. Methods* **19** 11–12
[30] Ho J, Kalchbrenner N, Weissenborn D and Salimans T 2019 arXiv:1912.12180
[31] Dosovitskiy A *et al* 2020 arXiv:2010.11929

[32] Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S and Guo B 2021 Swin transformer: hierarchical vision transformer using shifted windows *Proc. IEEE/CVF Int. Conf. on Computer Vision* pp 10012–22
[33] Santos J E, Fox Z R, Mohan A, O'Malley D, Viswanathan H and Lubbers N 2023 *Nat. Mach. Intell.* **5** 1–9
[34] Santos J E, Fox Z, Mohan A, Viswanathan H and Lubbers N 2022 *NeurIPS Machine Learning and the Physical Sciences Workshop*
[35] San O, Maulik R and Ahmed M 2019 *Commun. Nonlinear Sci. Numer. Simul.* **77** 271–87
[36] Jaegle A *et al* 2021 arXiv:2107.14795
[37] Singhal K *et al* 2022 arXiv:2212.13138
[38] Marcato A, Santos J E, Liu C, Boccardo G, Marchisio D and Franco A A 2023 *Energy Storage Mater.* **63** 102927
[39] Kingma D P and Ba J 2014 arXiv:1412.6980
[40] Colonius T and Taira K 2008 *Comput. Methods Appl. Mech. Eng.* **197** 2131–46
[41] NOAA National oceanic and atmospheric administration: Physical sciences laboratory (available at: https://psl.noaa.gov/)
[42] Paszke A *et al* 2019 *Advances in Neural Information Processing Systems* vol 32
[43] Harris C R *et al* 2020 *Nature* **585** 357–62
[44] Hunter J D 2007 *Comput. Sci. Eng.* **9** 90–95