A Multi-branch Decoder Network Approach to Adaptive Temporal Data Selection and Reconstruction for Big Scientific Simulation Data

Yang Zhang, *IEEE Student Member*, Hanqi Guo, *IEEE Member*, Lanyu Shang, *IEEE Student Member*, Dong Wang, *IEEE Member* and Tom Peterka, *IEEE Member*

Abstract—A key challenge in scientific simulation is that the simulation outputs often require intensive I/O and storage space to store the results for effective post hoc analysis. This paper focuses on a *quality-aware adaptive temporal data selection and reconstruction* problem where the goal is to adaptively select simulation data samples at certain key timesteps in situ and reconstruct the discarded samples with quality assurance during post hoc analysis. This problem is motivated by the limitation of current solutions that a significant amount of simulation data samples are either discarded or aggregated during the sampling process, leading to inaccurate modeling of the simulated phenomena. Two unique challenges exist: 1) the sampling decisions have to be made in situ and adapted to the dynamics of the complex scientific simulation data; 2) the reconstruction error must be strictly bounded to meet the application requirement. To address the above challenges, we develop *DeepSample*, an error-controlled convolutional neural network framework, that jointly integrates a set of coherent multi-branch deep decoders to effectively reconstruct the simulation data with rigorous quality assurance. The results on two real-world scientific simulation applications show that DeepSample significantly outperforms other state-of-the-art methods on both sampling efficiency and reconstructed simulation data quality.

Index Terms—Big Scientific Simulation Data, Adaptive Temporal Data Selection and Reconstruction, Multi-branch Decoder Network

1 Introduction

With the unprecedented computational power of supercomputers and advanced simulation modeling techniques, scientific simulation is a pivotal research paradigm to study the complex physical phenomena across engineering and scientific disciplines [1]. Examples of such applications include the air pollution simulation for greenhouse gas emission modeling [2], high energy cosmic ray simulation for cosmology studies [3], and fluid dynamics simulation for cardiovascular medicine research [4]. A key challenge in scientific simulation is that the outputs of the simulation experiments are often time-varying and high-dimensional data with *high-velocity* and *large-volume* [5], which require extensive resources (e.g., I/O bandwidth and storage space) to store sufficient timesteps for accurate post hoc analysis [6]. For example, in a high-fidelity climate simulation, an average of 260 TB time-varying 3D simulation data is generated by the simulation model every 16 seconds [7]. As a result, the climate simulation model requires an average of 16.25 TB/s I/O speed to support this application. However, the peak output rate for the current parallel file system in high-performance computing (HPC) systems is only 2.5 TB/s [8]. Therefore, such a large gap between the intensive data requirement from scientific simulations and the I/O bottleneck of the current HPC systems presents a critical challenge for the big data community.

Previous efforts have been made to address the I/O and storage bottleneck in scientific simulations. Common practices include uniform sampling and averaging [9], stratified random sampling [10], content-aware adaptive sampling [11], and bitmap indexing [12]. However, a key limitation of these solutions is that a significant amount of simulation timesteps are either discarded or aggregated during the sampling process, which could hinder the scientists from building accurate models for the studied phenomena in the post hoc analysis [13]. There also exist some recent efforts in lossy compression that compress the data sample at each timestep to reduce the I/O overhead [5], [14]. However, those solutions mainly focus on reducing the data size at the *spatial* dimension and miss the opportunity to reduce the data size in the temporal dimension. As a result, they could be insufficient to meet the I/O requirements of the long-term and large-scale scientific simulation applications.

In this paper, we study the in situ *quality-aware adaptive temporal data selection* problem in scientific simulations. In particular, our problem aims to adaptively select samples from the streaming simulation data at certain timesteps (i.e., key timesteps) in situ and reconstruct the discarded samples during post hoc analysis to meet certain quality requirements (e.g., quality bounds) of the application. The key timesteps need to be a small fraction of the total timesteps in the simulation process to reduce the requirements on I/O bandwidth and storage space. Meanwhile, the reconstructed data quality from the selected data samples has to ensure the

- *Y. Zhang is with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA, 46556. E-mail: yzhang42@nd.edu*
- *L. Shang, and D. Wang are with the School of Information Sciences, University of Illinois Urbana-Champaign, Champaign, Champaign, IL, USA, 61820. E-mail: lshang3@illinois.edu, dwang24@illinois.edu*
- *H. Guo and T. Peterka are with the Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA, 60439. E-mail: hguo@anl.gov, tpeterka@mcs.anl.gov*
- *Corresponding author: D. Wang (dwang24@illinois.edu)*

expected post hoc analysis performance of the simulated scientific applications [14]. Two unique challenges exist in solving the quality-aware adaptive temporal data selection problem, which are elaborated as follows.

In situ Adaptive Sampling for Complex Simulation Data: The scientific simulation data are known to be high-dimensional and large-volume with nonlinear and complex temporal correlations [5]. The key question here is: how can we identify the key timesteps for sampling in situ to achieve the objective of maximizing the number of discarded samples while ensuring the reconstructed data quality? The current solutions in scientific simulations often require the access to a large volume of data samples to learn the complex temporal correlations to address the above question [15], [16]. However, those approaches introduce a non-trivial I/O and storage overhead which is not affordable for the big data scientific simulations we studied in this paper. The adaptive sampling technique for streaming data processing could be applied to address the above challenge [17], [18]. However, such a technique is often designed for specific signal and data processing tasks (e.g., streaming data summarization, entropy-driven checkpoint detection) and is difficult to be adapted to the high-dimensional massive simulation data samples. Therefore, an in situ adaptive data selection scheme that can effectively sample and reconstruct the complex streaming simulation data is yet to be developed.

Error-controlled Non-uniform Data Reconstruction: The second challenge lies in ensuring an error-controlled data reconstruction from the data samples that are adaptively (non-uniformly) sampled at the key timesteps. Such an error-controlled data reconstruction is essential for the accurate modeling in post hoc analysis [19]. However, two important limitations exist in the current solutions to address such a challenge. First, current data reconstruction solutions in scientific simulations primarily focus on optimizing the reconstructed data quality without a rigorous guarantee (e.g., performance bounds) on the reconstructed data quality [13], [20]. Second, current solutions in temporal scientific simulation data reconstruction are mainly designed based on the assumption that the simulation data is uniformly sampled [21]. This is because those solutions require a fixed sampling window to determine the structure of the neural networks before the network optimization process [22]. Such a limitation makes those approaches incapable of recovering the simulation data sequence using samples from the non-uniformly distributed key timesteps that are critical to accommodate the dynamics and provide quality assurance for the reconstructed data.

To address the above challenges, we develop *DeepSample*, an error-controlled deep data sampling and reconstruction scheme to solve the in situ quality-aware adaptive temporal data selection problem in scientific simulations. In particular, we develop a quality-aware data sampling framework that makes online sampling decisions to adaptively select data samples from the key timesteps by leveraging the estimated data reconstruction quality from a novel deep estimation network. Furthermore, we develop an error-controlled convolutional neural network architecture that integrates a set of coherent multi-branch deep decoders to effectively reconstruct the discarded data samples with rigorous data quality bounds. To the best of our knowledge, DeepSample is the first *error-controlled* deep learning approach to solve the adaptive temporal data sampling and reconstruction problem with quality assurance in scientific simulations. We evaluate DeepSample through two real-world scientific simulation applications (i.e., Shear Boundary Layer and Shallow Cumulus Convection). The evaluation results show that our DeepSample consistently outperforms the state-of-the-art baselines by simultaneously achieving better sampling efficiency and reconstructed data quality under various application scenarios.

2 Related Work

2.1 Scientific Simulation Data Management

Driven by the advent of recent high-performance computing and simulation modeling techniques, scientific simulation has been established as a powerful and scalable paradigm to study the complex real-world phenomena in many scientific and engineering disciplines [1]. Our paper focuses on the effective simulation data management and analysis in large-scale scientific simulations [23]. Examples of those applications include flexible I/O design for heterogeneous runtime platforms [24], fast lossy compression and recovery of high throughput simulations [25], and in situ visualization of ultra-scale scientific simulations [9]. Several key challenges exist in these applications. Examples include high data velocity and dimensionality, model portability, and system scalability [26]. However, the quality-aware adaptive data selection problem remains to be an open problem in scientific simulation data management and analysis. In this paper, we develop a novel error-controlled deep data sampling and reconstruction framework to address this problem by adaptively sampling the simulation data at key timesteps to effectively reduce the I/O and storage overhead while ensuring the desirable reconstructed data quality.

2.2 Data Sampling and Reconstruction for Scientific Simulations

Previous efforts have been made to address the data sampling and reconstruction problem in scientific simulation applications [5], [13], [14], [15], [16], [20]. Examples of those solutions include uniform data sampling and reconstruction [13], [20], adaptive key step selection [15], [16], and lossy compression [5], [14]. For example, Han *et al.* developed a 3D deep temporal super-resolution approach to reconstruct the time-varying simulation data sequence from the uniform samples using recurrent neural networks [13]. Zhou *et al.* proposed a deep data reconstruction framework to preserve the structural integrity of the simulation outputs during the data reconstruction process using convolution neural networks [20]. However, those data reconstruction approaches can not be adopted to work with the non-uniform samples that are essential to provide quality assurance of the reconstructed data [22]. On the other hand, there exist several efforts on adaptive key timestep selection in scientific simulations [15], [16]. For example, Tong *et al.* proposed a dynamical temporal sample framework to retrieve the most salient timesteps from large-scale time-varying simulation datasets using dynamic time warping [15]. However, those adaptive sampling approaches require to access

all simulation data to effectively identify the key timesteps and introduce a non-trivial I/O and storage overhead. There also exist a couple of recent efforts in lossy compression that are designed to reduce the simulation data size at each timestep to reduce the I/O overhead [5], [14]. However, those lossy compression approaches mainly focus on reducing the data size at the spatial dimension but miss the opportunity to reduce the data size in the temporal dimension. As a result, they could be insufficient to reduce the I/O overhead for the long-term and large-scale scientific simulations. In contrast, our paper focuses on adaptively selecting a small fraction of samples from the streaming simulation data in situ and effectively reconstructing the discarded samples during post hoc analysis to meet the data quality requirements of the application.

2.3 Deep Learning for Scientific Simulation Data

Our work is also related to the growing trend of utilizing deep learning techniques in scientific simulation data management and analysis including particle tracing, streaming lines clustering, noise reduction, and parameter space exploration [6], [27], [28], [29]. For example, Han *et al.* proposed a deep clustering approach to select the streamlines and stream surfaces for effective flow visualization using decoder-encoder architectures [27]. Hong *et al.* developed an access pattern estimation scheme to trace the parallel particle in flow field using long short-term memory networks [28]. Kim *et al.* designed a reference frame extraction framework to reduce the noise and artifacts in unsteady vector fields using convolutional neural networks [29]. He *et al.* developed a deep image synthesis approach to explore the parameter space for simulation data generation using generative adversarial networks [6]. To the best of our knowledge, DeepSample is the first error-controlled multi-branch deep decoder network approach to solve the adaptive data sampling and reconstruction problem with quality assurance in scientific simulations.

3 Problem Definition

In this section, we formally define the quality-aware adaptive temporal data selection problem in scientific simulations.

Definition 1. **Simulation Data ($\mathcal{D}$)**: We define $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_T\}$ to represent the simulation data (e.g., 3D volume data of weather simulations) generated by a scientific simulation model, where D_t indicates the simulation data generated at timestep t. T is the total number of timesteps in the scientific simulation application.

Definition 2. **Selected Set ($\mathcal{S}$)**: We define $\mathcal{S}$ to represent the set of simulation data selected from the entire simulation data sequence $\mathcal{D}$, which is used to reconstruct the discarded simulation data samples during the post hoc analysis. In particular, we define $\mathcal{S} = \{\mathcal{D}_1^{\mathcal{S}}, \mathcal{D}_2^{\mathcal{S}}, ..., \mathcal{D}_I^{\mathcal{S}}\}$. I is the total number of simulation data samples in $\mathcal{S}$. Mathematically, we have:

$$\mathcal{S} \subset \mathcal{D}, \text{where } I < T \tag{1}$$

Definition 3. **Discarded Set ($\mathcal{X}$)**: We defined $\mathcal{X}$ to represent the simulation data discarded during the in situ selection process. In particular, we define $\mathcal{X} = \{\mathcal{D}_1^{\mathcal{X}}, \mathcal{D}_2^{\mathcal{X}}, ..., \mathcal{D}_J^{\mathcal{X}}\}$. J is the number of discarded data samples in $\mathcal{X}$. Mathematically, we have:

$$\mathcal{D} = \mathcal{X} \cup \mathcal{S}, \text{where } T = I + J \tag{2}$$

Definition 4. **Reconstructed Set ($\widehat{\mathcal{X}}$)**: We define $\widehat{\mathcal{X}}$ to represent the reconstructed simulation data for all simulation data discarded in the discarded set $\mathcal{X}$. In particular, we define $\widehat{\mathcal{X}} = \{\widehat{\mathcal{D}_1^{\mathcal{X}}}, \widehat{\mathcal{D}_2^{\mathcal{X}}}, ..., \widehat{\mathcal{D}_J^{\mathcal{X}}}\}$, where $\widehat{\mathcal{D}_j^{\mathcal{X}}}$ indicates the reconstructed simulation data for $\mathcal{D}_j^{\mathcal{X}}$. Note that the size of the reconstructed set $\widehat{\mathcal{X}}$ is the same as the size of the discarded set $\mathcal{X}$.

Definition 5. **Reconstruction Quality Bound (θ)**: We define θ to represent the reconstruction quality bound for a specific scientific simulation application (e.g., a predefined peak signal-to-noise ratio (PSNR) or structural similarity index measure (SSIM) [30] threshold that meets the specific application requirement for the post hoc analysis). Our goal is to ensure the reconstructed data quality strictly meets the reconstruction quality bound θ during the data reconstruction process.

Definition 6. **Discard Ratio (α)**: We define α to represent the ratio for the number of simulation data samples in the discarded set $\mathcal{X}$ over the number of simulation data samples at all time steps $\mathcal{D}$ as follows:

$$\alpha = \frac{|\mathcal{X}|}{|\mathcal{D}|} = \frac{J}{T} \tag{3}$$

Given the above definitions, the goal of our adaptive temporal data selection problem is to dynamically select a subset of simulation data samples which can be used to effectively reconstruct the discarded samples that meet the reconstruction quality bound. Using the above definitions, our problem is formally defined as:

$$\begin{aligned} &\arg\max_{\mathcal{S}}(\alpha|\mathcal{D}, \theta), \text{ while } \mathcal{F}(\widehat{\mathcal{X}}, \mathcal{X}) \geq \theta \\ &\text{where } \mathcal{S} \subset \mathcal{D} \text{ and } |\mathcal{S}| < |\mathcal{D}| \end{aligned} \tag{4}$$

where $\mathcal{F}$ is the function to evaluate the quality of the reconstructed simulation data (e.g., PSNR and SSIM). This problem is challenging considering that the adaptive data selection decisions have to be made in situ given the complex high-dimensional, large-volume and time-varying scientific simulation data and the strict reconstruction quality bound requirement to the reconstructed simulation data quality. In this paper, we develop the DeepSample scheme to address these challenges, which is elaborated next.

4 Solution

In this section, we present the DeepSample framework to address the quality-aware adaptive temporal data selection problem in scientific simulation applications. We first present an overview of the framework and then discuss its components in detail. In the end, we summarize the DeepSample framework in pseudocode.

4.1 Overview of DeepSample Framework

DeepSample is an error-controlled multi-branch convolutional neural network framework to adaptively identify the key timesteps in scientific simulation outputs that can be used to effectively reconstruct the discarded data samples with desirable quality assurance. An overview of DeepSample is shown in Figure 1. It consists of two major modules:

- *Error-Controlled Adaptive Sampling (ECAS)*: it designs an error-controlled adaptive sampling mechanism that performs in situ data sampling to dynamically select the simulation data on demand to meet the reconstruction quality requirement of the application.
- *Multi-Branch Deep Data Reconstruction (MDDR)*: it develops a multi-branch deep convolutional neural network architecture to effectively reconstruct the discarded data samples during post hoc analysis through a novel *one-to-many* deep encoder-decoder design.

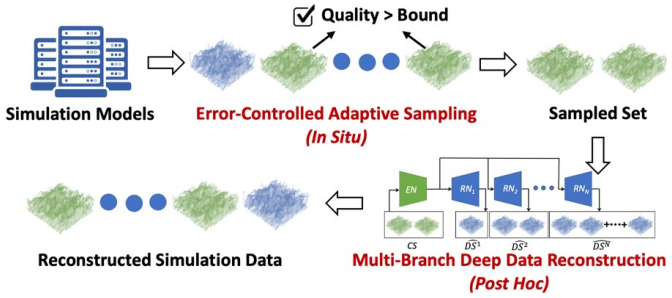


Figure 1. Overview of the DeepSample Framework

4.2 Error-Controlled Adaptive Sampling

In this subsection, we present the ECAS module in DeepSample to adaptively select the data samples in order to meet the expected reconstruction quality bound specified by the application. We first start with a few key definitions that will be used in our ECAS module.

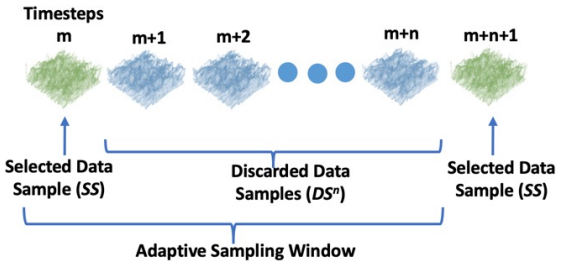


Figure 2. Illustrations of the Key Sampling Concepts

In particular, the ECAS module aims to adaptively identify the sampling window (i.e., a series of data samples between two consecutive data selection operations (see Figure 2)), where ECAS only selects the data samples at the starting timestep of each sampling window (i.e., D_m) into the selected set $\mathcal{S}$ (Definition 2) and discards all data samples at the intermediate steps (i.e., $\{\mathcal{D}_{m+1}, \mathcal{D}_{m+2}..., \mathcal{D}_{m+n}\}$) to reduce I/O bandwidth and storage overhead. At the post hoc stage, DeepSample reconstructs $\{\mathcal{D}_{m+1}, \mathcal{D}_{m+2}..., \mathcal{D}_{m+n}\}$ using D_m and $\mathcal{D}_{m+n+1}$ from the starting timesteps of two consecutive sampling windows. In particular, our adaptive sampling scheme makes the sampling decision by maximizing the number of discarded samples in the intermediate timesteps (i.e., minimizing the I/O bandwidth and storage overhead) under the reconstruction quality bounds for each sampling window. We will provide details on how to dynamically identify the sampling window in the rest of this subsection. Please also note that the data samples at the first and last timesteps of a simulation application are selected into the selected set $\mathcal{S}$ by default.

Definition 7. **Selected Data Sample** (SS): We define SS to represent the data sample selected at the sampling timestep in a sampling window (e.g., $\mathcal{D}_m$).

Definition 8. **Consecutively Selected Data Samples** (CS): We define CS to represent a pair of consecutively selected data samples (e.g., both $\mathcal{D}_m$ and $\mathcal{D}_{m+n+1}$).

Definition 9. **Discarded Data Samples (DS^n)** : We define DS^n to represent the n data samples between two consecutive data sampling timesteps that are discarded during the sampling process (e.g., $\{\mathcal{D}_{m+1}, \mathcal{D}_{m+2}..., \mathcal{D}_{m+n}\}$)

We further define a temporal data reconstruction function as follows:

Definition 10. **Temporal Data Reconstruction Function ($\mathcal{R}$)**: We define $\mathcal{R}$ to be a temporal data reconstruction function that reconstructs the discarded data samples at n intermediate timesteps using the selected data samples from two consecutive sampling timesteps as follows:

$$\widehat{DS^n} = \{\widehat{\mathcal{D}_{m+1}}, \widehat{\mathcal{D}_{m+2}}..., \widehat{\mathcal{D}_{m+n}}\} = \mathcal{R}(\mathcal{D}_m, \mathcal{D}_{m+n+1}) \tag{5}$$

where $\widehat{DS^n}$ represents the reconstructed data samples at the intermediate timesteps. For example, $\widehat{\mathcal{D}_{m+1}}$ indicates the reconstructed sample at timestep $m+1$. We will discuss the detailed design of R that maximizes the reconstructed data quality in the next subsection.

With $\widehat{DS^n}$, we can compute the reconstructed data quality as follows:

$$\mathcal{F}(\widehat{DS^n}, DS^n) \tag{6}$$

where $\mathcal{F}$ represents the quality metric (e.g., PSNR, SSIM). If the reconstructed data quality $\mathcal{F}(\widehat{DS^n}, DS^n)$ meets the quality bound θ (defined in Definition 5), i.e., $\mathcal{F}(\widehat{DS^n}, DS^n) \geq \theta$, the ECAS module only needs to save the data samples at the sampling timesteps and discard all the samples at the intermediate timesteps to save the I/O bandwidth and storage space. In particular, our adaptive sampling scheme makes the sampling decision by maximizing the number of discarded samples in the intermediate timesteps (i.e., n) to meet the reconstruction quality bounds:

$$\begin{aligned} \underset{\mathcal{D}_m, \mathcal{D}_{m+n+1}}{\arg\max} \ & (n | \mathcal{F}(\mathcal{R}(\mathcal{D}_m, \mathcal{D}_{m+n+1}), DS^n) > \theta) \\ & \text{add } \mathcal{D}_m, \mathcal{D}_{m+n+1} \text{ to } \mathcal{S} \\ & \text{discard } \{\mathcal{D}_{m+1}, \mathcal{D}_{m+2}, ..., \mathcal{D}_{m+n}\} \end{aligned} \tag{7}$$

where $\mathcal{S}$ is the sampled set during the adaptive sampling process (Definition 2). We observe that the discard ratio

of our scheme depends heavily on the temporal data reconstruction function $\mathcal{R}$. For example, if $\mathcal{R}$ can accurately reconstruct the discarded data samples even when n is large, our scheme can achieve a high discard ratio. In addition, $\mathcal{R}$ should also be able to work with sampling windows of varying sizes in order to support the adaptive sampling decisions. We elaborate the design details of such an effective temporal data reconstruction function $\mathcal{R}$ below.

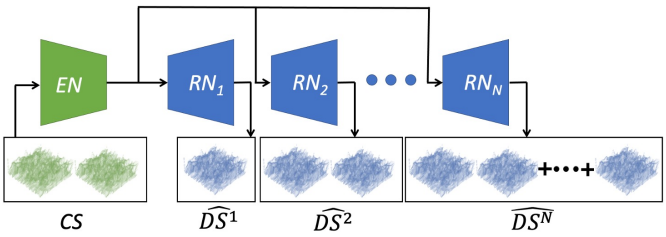


Figure 3. Overview of *One-to-Many* Network Design

4.3 Multi-Branch Deep Data Reconstruction

In this section, we present the detailed design of the temporal data reconstruction function $\mathcal{R}$ we introduced in the ECAS module. In particular, we design a multi-branch deep reconstruction network in DeepSample to effectively reconstruct all discarded data samples. Our MDDR design aims to address a key limitation of the current deep temporal data reconstruction solutions in scientific simulations: they can only reconstruct simulation data when the data is *uniformly* sampled (i.e., the length of the sampling window is fixed). This is because those solutions require the fixed size of the sampling window to determine the structure of the neural networks before the network optimization process. In particular, the learned weights in the optimized network instances become invalid if the size of the sampling window changes after the network optimization process. Such a limitation makes those approaches incapable of dynamically maximizing the number of discarded data samples (i.e., n) to effectively reduce the I/O bandwidth and storage overhead.

Our MDDR module is designed to address the above limitation. In particular, the MDDR consists of two types of neural networks: an extraction network (EN) and a set of reconstruction networks (RN). The overall architecture of the multi-branch network design is shown in Figure 3. In our MDDR module, the extraction network EN and the reconstruction network RNs work collaboratively to learn a deep data reconstruction model that can concurrently reconstruct the discarded intermediate data samples given different sizes of sampling windows. In particular, the EN first extracts both high-level (e.g., objects and patterns) and low-level (e.g., colors and textures) visual features from the simulation data samples at two consecutive sampling timesteps. The EN captures the complex non-linear temporal evolution in the simulation data sequence. Then each RN_n explicitly fuses the visual features and temporal evolution extracted by EN to reconstruct the discarded data samples from sampling windows with different sizes (e.g., RN_n reconstructs the discarded data samples for n intermediate timesteps). To the best of our knowledge, the MDDR is the first multi-branch deep reconstruction architecture that introduces a novel *one-to-many* deep encoder-decoder design to simultaneously reconstruct the discarded data samples with time varying sampling window sizes. We first formally define the extraction network EN and the reconstruction network RN as follows:

Definition 11. **Extraction Network (*EN*)**: We define EN as a mapping network to extract visual features (e.g., objects and patterns, color and texture distributions) from the simulation data at two consecutive sampling timesteps and capture the complex temporal evolution and non-linear dynamics in the data sequences as follows:

$$V^{CS} = EN(CS) \tag{8}$$

where V^{CS} is used to represent the extracted visual features and temporal evolution from the selected data samples at the consecutive sampling timesteps.

We present the detailed layer-wise architecture design of EN in the (A) of Figure 4. It contains a stack of convolutional layers connected with a set of residual blocks that provide sufficient network depth to extract the visual features and capture the complex evolution of the simulation data from the consecutively selected data samples. In addition, we also design a set of loss functions to ensure that the EN is capable of accurately capturing the visual features and complex evolution. We also enable the skip connection to the extraction network (i.e., the dotted lines in Figure 4), which is designed to forward different levels of visual features (e.g., high-level visual features including objects and patterns, and low-level features including colors and textures) extracted by EN to RN. The different levels of visual features then can be utilized by RN to reconstruct the discarded data samples at the intermediate timesteps.

Definition 12. **Reconstruction Network (*RN*)**: We define RN_n as a reconstruction network that reconstructs the discarded data at intermediate timesteps using the visual features V^{CS} extracted by EN:

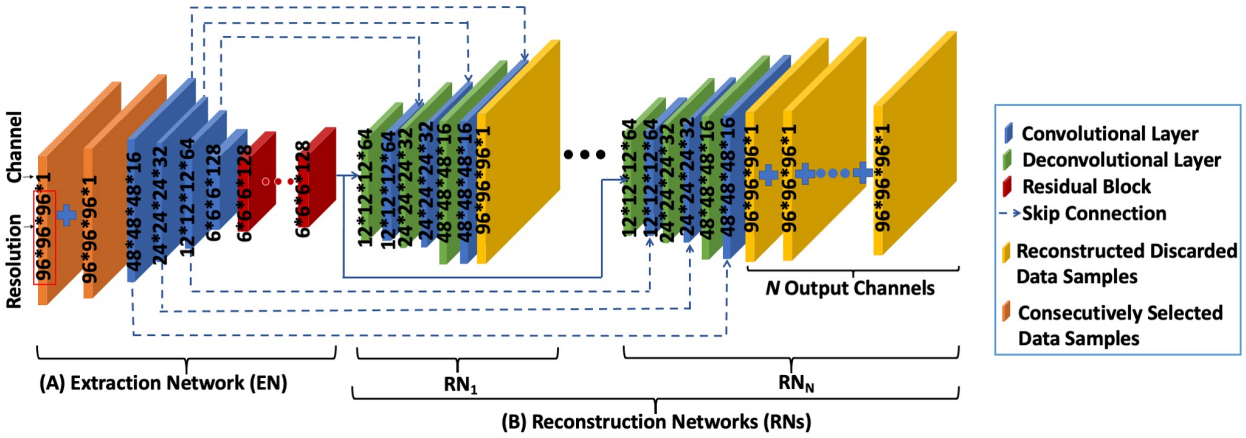
$$\widehat{DS^n} = RN_n(V^{CS}) \tag{9}$$

where $\widehat{DS^n}$ is the reconstructed data samples for n intermediate timesteps.

We present the detailed layer-wise architecture design of RN in (B) of Figure 4. In particular, the reconstruction network consists of a set of deconvolution layers that reconstruct the simulation data by gradually adding the fine-grained visual features at different levels to the reconstructed data samples. In addition, RN also includes a set of convolution layers that fuse different levels of visual features extracted by EN through skip-connections. This is done to ensure all levels of visual features are successfully preserved in the reconstructed data samples at all intermediate timesteps.

Our next question is how to learn the optimal instances of all networks in DeepSample to jointly maximize the reconstructed data quality with the dynamic sampling window. To address this question, we define the reconstruction loss $\mathcal{L}_{rec}^{EN,RN_n}$ for EN with RN_n as follows:

$$\mathcal{L}_{rec}^{EN,RN_n} : \mathcal{L}_{\text{voxel}}(DS^n, RN_n(EN(CS))) \tag{10}$$



Channel indicates the number of dimensions representing the visual information embedded at each voxel (e.g., a channel represents the vertical thermodynamic structure of the atmosphere). *Resolution* indicates the spatial size of the matrix that represents the visual information preserved in each neural work layer.

Figure 4. Overall Network Architecture of MDDR

where CS represents the selected data samples at two consecutive sampling timesteps. $\mathcal{L}_{\text{voxel}}$ indicates the voxel-wise mean square error (MSE) loss [31] that measures the voxel-wise value differences between the original and reconstructed simulation data samples. Intuitively, $\mathcal{L}_{rec}^{EN,RN_n}$ is designed to ensure the stable performance of EN and RN_n in reconstructing the high quality discarded data samples. Finally, we combine the reconstruction loss functions for all reconstruction networks to derive the final loss $\mathcal{L}_{final}^{EN,RN}$ for the MDDR module as follows:

$$\mathcal{L}_{final}^{EN,RN} : \sum_{n=1}^{N} \mathcal{L}_{\text{voxel}}(DS^n, RN_n(EN(CS))) \tag{11}$$

where N indicates the number of reconstruction networks in the MDDR module. Ideally, N is set to be the size of the whole simulation data to cover discarded data samples with all possible sizes. However, it is infeasible to maintain such a large number of reconstruction networks in practice. Therefore, the end users of our model need to set the maximum number of reconstruction networks (i.e., N) based on the available memory space to maintain the MDDR network [32]. In practice, we recommend setting a larger N if the memory space permits, since it allows more data samples to be potentially discarded in each sampling window and could potentially lead to a higher discarded ratio. We also add a parameter study in Section 5.3.6 to study the performance of our DeepSample scheme by varying the number of reconstruction networks.

Using the above loss function, we can learn the optimal instances (i.e., EN^*, RN_n^*) of all networks using the Adaptive Moment Estimation (ADAM) optimizer [33]. Finally, we use EN^* and RN_n^* to reconstruct the data samples with the dynamic sampling windows as follows:

$$\widehat{DS^n} = RN_n^*(EN^*(CS)) \tag{12}$$

Our MDDR design guarantees the reconstruction quality bound θ through an iterative quality assurance process during the in situ sampling stage. In particular, we use every reconstruction branch of the pre-trained network instances (EN^*, RN_n^*) to examine the reconstruction data quality in order to make a sampling decision (i.e., a data sample is selected or discarded). In the best-case scenario, the reconstruction network RN_n^* with the longest reconstruction timestep interval from all reconstruction networks in MDDR (i.e., RN_n^* with the largest n) can reconstruct intermediate timesteps while meeting the designated quality bound θ. In the worst-case scenario where none of the reconstruction network outputs meets the quality bound θ, our scheme does not discard any timesteps. In such a case, our scheme does not need to reconstruct any discarded data, thus the quality bound θ is still satisfied. In all cases, the quality bound is guaranteed when we reconstruct the discarded data samples using the same network instances (EN^*, RN_n^*) during the post hoc reconstruction stage as the ones we used to examine the reconstruction quality during the in situ sampling stage.

4.4 Summary of DeepSample Framework

Finally, we further summarize the detailed execution steps of DeepSample in Algorithm 1. In particular, DeepSample includes three main phases in performing the in situ adaptive temporal data sampling and reconstruction as follows:

Model training phase: The objective is to train an optimized multi-branch deep data reconstruction network (EN^* and RN_n^*) in MDDR that will be used for both in situ sampling and post hoc reconstruction. Following the deep model training procedure of scientific data [6], the training data can be obtained from a pre-run of the simulation model with different simulation parameter settings.

In situ sampling phase: Given the learned optimized EN^* and RN_n^*, our next objective is to adaptively identify the selected set S from the simulation outputs D. In particular, the EN^* and RN_n^* are used as the temporal data reconstruction function (Definition 10) in the ECAS module for adaptive sampling decisions. Note that our DeepSample does not involve any network training during the sampling phase. Instead, it utilizes the learned network instances (EN^* and RN_n^*) learned from the model training phase to make the

real-time sampling decision, where the sampling decision only requires constant time at each timestep given a specific scientific simulation application [34]. In addition, the time complexity of the sample decision at each timestep only grows linearly with respect to the size of input simulation data samples for different scientific simulation applications. This is because we will only need to utilize the learned network instances as the temporal data reconstruction function to check the reconstructed data quality at each time step [35].

Post hoc reconstruction phase: Given the identified selected set S, our objective in the post hoc reconstruction phase is to effectively reconstruct the discarded data samples $\widehat{\mathcal{X}}$, where the EN^* and RN_n^* use the sampled set to reconstruct the discarded data samples to meet the quality requirements from the application.

In practice, our model is pre-trained with archived simulation outputs (referred to as *pre-run* data). The pre-trained model is then applied to the simulation data that share similar basic characteristics with the pre-run data. For a domain scientist, the niche application of our method is to reduce output data in *ensemble simulations*, which play a vital role in many engineering and scientific disciplines including weather research, fluid dynamics, and cosmology [1]. In particular, ensemble simulations require generating simulation data from the same simulation with different physical parameter variations [6]. In this case, our model only needs to be trained with the pre-run data generated by the simulation (e.g., weather simulations) using a single set of parameter settings. The pre-trained model can enable intelligent data-reduction towards the data generated by the same simulation with many different physical parameter variations. Note that one should not expect a high discarded ratio for different types of simulations such as fluid dynamics or cosmology using the model for weather simulations because data characteristics are much different [36]. In addition, our model can also be applied by training with the data generated at the beginning of the simulation. For example, our model could be applied to enable intelligent data-reduction in the simulations that exhibit periodical physical behaviors (e.g., surface vortex generation and diminishing) where we can train our model in the first few periods of the simulation and apply it to the following periods. On the other hand, we also note that the adaptive sampling performance of our scheme might not be optimized if the physical behavior of the studied simulation changes dramatically after the beginning stage. This is because the model learned at the beginning could be overfitted to the simulation data generated later with significantly different physical behaviors. A potential solution to address this issue is to periodically retrain our model offline using newly generated data to learn the changed physical behaviors. We then replace the old model with the retrained one once the retaining process is done to ensure the desirable adaptive sampling performance.

Algorithm 1 DeepSample Framework Summary

▷ *model training phase*
initialize EN (Definition 11)
initialize all RN_n (Definition 12)
for each epoch **do**
  for each batch **do**
    optimize EN and all RN_n (Equation (11))
  end for
end for
obtain EN^* and all RN_n^*
▷ *in situ sampling phase*
set EN^* and all RN_n^* as $\mathcal{R}$ (Definition 10)
for t from *1* to T **do**
  identify SS from D (Equation (7))
  add SS to $\mathcal{S}$
  discard DS
end for
output $\mathcal{S}$
▷ *post hoc reconstruction phase*
for all *CS* in S **do**
  reconstruct $\widehat{DS}$ from *CS* using EN^* and all RN_n^* (Equation (12))
  add $\widehat{DS}$ to $\widehat{\mathcal{X}}$
end for
output $\widehat{\mathcal{X}}$

5 Evaluation

In this section, we conduct extensive experiments using the simulation data collected from two real-world scientific simulation applications to address the questions below:

- Q1: Can DeepSample outperform the state-of-the-art baselines in terms of reconstructed data quality?
- Q2: How does DeepSample perform on the discard ratio compared to the baselines?
- Q3: Can DeepSample meet the quality bounds of the application with a high discard ratio?
- Q4: Can DeepSample help the spatial data reduction tools (e.g., lossy compressors) to further improve their performance?
- Q5: How does each component of DeepSample design contribute to its overall performance?
- Q6: How do the different choices of model parameters (e.g., the number of reconstruction networks) affect the performance of DeepSample?

5.1 Dataset

We evaluate DeepSample using the publicly available simulation platform provided by CM1 [1] to generate the simulation datasets from two real-world scientific simulation applications: *Shear Boundary Layer* and *Shallow Cumulus Clouds*. We choose these two simulation applications because they provide the high-dimensional, non-linear, and time-varying simulation outputs that create challenging evaluation scenarios for our DeepSample scheme. In particular, we show the dynamic evolving characteristics of the evaluation datasets by plotting the mean absolute difference for the data samples between two consecutive timesteps in Figure 5. We summarize the datasets as follows:

Shear Boundary Layer: Shear Boundary Layer is an important computational fluid dynamic simulation application, which studies the complex physical phenomenon of fluid in the immediate vicinity of a bounding surface with significant viscosity effects [37]. We focus on the complex flow velocity (i.e., three components of flow velocity along x, y, z-axis) of the boundary layers generated in this simulation. The simulation data sample at each timestep is in a dimension of $96 \times 96 \times 96 \times 3$.

1. https://www2.mmm.ucar.edu/people/bryan/cm1/

Shallow Cumulus Convection: Shallow cumulus convection is a critical atmosphere physics simulation application, which studies the vertical thermodynamic structure of the atmosphere [38]. We focus on the convection in the vertical direction in this study. The simulation data sample at each timestep is in a dimension of $64 \times 64 \times 64 \times 1$.

In our experiments, we use 100 data samples from an ensemble run as the training data and use additional 500 data samples from a different ensemble run as the testing data for both datasets. In particular, we observe both simulated applications exhibit periodical physical behaviors (e.g., surface vortex generation and diminishing) over time. We test the model with a 500-timestep interval, which covers a sufficient number of periods on both datasets to obtain reliable results for evaluation.

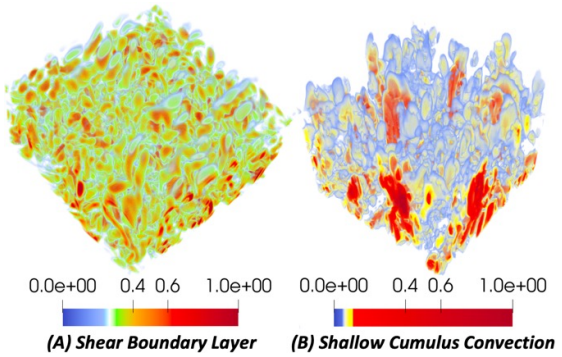


Figure 5. Mean Absolute Difference for Data Samples between Two Consecutive Timesteps

5.2 Baselines and Metrics

- *Lerp [39]*: a widely used conventional data reconstruction scheme that utilizes linear interpolation to reconstruct discarded data samples with varying sizes. In particular, we use Lerp to replace the MDDR module as the temporal data reconstruction function in our ECAS module, which allows Lerp to make adaptive sampling decisions to meet the corresponding reconstruction quality bounds.
- *UNet [40]*: a recent deep neural network architecture that utilizes the contracting paths to learn multi-level temporal dynamics from the dataset to reconstruct the discarded samples. Following the standard UNet architecture, the UNet baseline includes an encoder for deep feature extraction and a decoder for data reconstruction. The encoder and decoder are also connected by three additional skip-connections. In addition, we follow [22] to extend UNet so that it can reconstruct the discarded data samples with varying sizes. In particular, we interpolate deep features extracted by the encoder from consecutively selected data samples to generate deep features for each discarded data sample from intermediate timesteps. We then forward the generated deep features to the decoder to reconstruct the discarded data samples with varying sizes. Similar to Lerp, we also use the UNet as the temporal data reconstruction function in our ECAS module to provide adaptive data sampling and reconstruction.
- *FCNN [41]*: a deep linear embedding model that utilizes the fully convolutional neural networks (FCNN) to map the simulation data into a latent deep feature space to interpolate the data samples at the intermediate timesteps. Similar to UNet, we also extend FCNN to enable error-controlled adaptive data sampling and reconstruction.
- *RNN [42]*: a representative deep learning baseline that utilizes long short-term memory (LSTM) networks to learn the dense spatial-temporal functions and recover the discarded data samples. It includes a convolutional layer for feature extraction from the selected data samples, a CovLSTM layer to learn the complex temporal evolution for the discarded data samples, and a convolutional layer for the final data reconstruction. The RNN baseline is designed to be coupled with the uniform sampling scheme to reconstruct the discarded data samples from uniformly sampled data samples.
- *TSR-TVD [13]*: a deep temporal super-resolution approach that reconstructs the time-varying simulation data sequence from the uniform samples using recurrent and convolutions neural networks. In particular, we follow the same network architecture as described in [13]. Similar to RNN, TSR-TVD is also designed to be coupled with the uniform sampling scheme for data sampling and reconstruction.

In our experiment, we adopt two representative evaluation metrics that are widely used to evaluate the reconstructed data quality in scientific simulations: *Peak Signal-to-Noise Ratio (PSNR)* and *Structural Similarity Index (SSIM)* [30]. In addition, we use the *discard ratio* (Definition 6) to evaluate the effectiveness of compared schemes in terms of reducing the I/O and storage overheads. In our experiment, all compared schemes are implemented using Pytorch (Version 1.1.0) [2] libraries and trained on the NVIDIA V100 GPUs. In our experiment, all hyper-parameters are optimized using the Adam optimizer [33]. In particular, we set the learning rate to be 10^{-4} and set the batch size to be 1. We also set the number of reconstruction networks N in our MDDR module to be 10. All compared schemes are trained over 500 epochs.

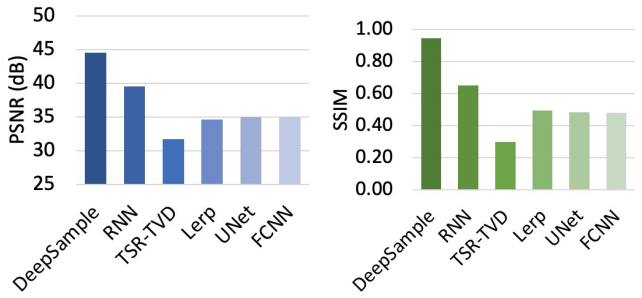
5.3 Evaluation Results

5.3.1 Q1: Data Reconstruction Quality

We first evaluate the multi-branch deep data reconstruction (MDDR) network design in our DeepSample model in terms of the data reconstruction quality. For a fair comparison, we keep the discard ratio of all schemes to be the same by fixing their sampling windows as 10. This ensures all compared schemes take the same amount of data samples in the reconstruction process. The evaluation results are shown in Figure 6 and Figure 7. We observe that DeepSample consistently outperforms all compared baselines on both datasets. For example, the performance gains achieved by DeepSample compared to the best-performing baseline (i.e.,

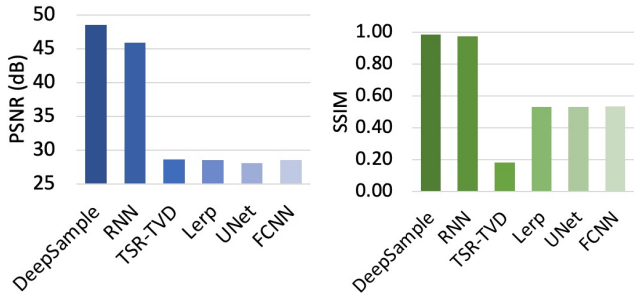
2. https://pytorch.org/

RNN) on the Shear Boundary Layer dataset on PSNR and SSIM are 5.03 and 0.292, respectively. We further visualize examples of the root mean absolute errors (RMSE) of the reconstructed data samples for all compared schemes in Figure 8 and Figure 9, respectively. We observe that our DeepSample achieves the best reconstructed data quality with the least RMSE. Such performance gains demonstrate our DeepSample model effectively utilizes the sampled data to maximize the reconstructed data quality.



(a) PSNR Comparison (b) SSIM Comparison

Figure 6. Performance Comparisons on Data Reconstruction Quality (Shear Boundary Layer)



(a) PSNR Comparison (b) SSIM Comparison

Figure 7. Performance Comparisons on Data Reconstruction Quality (Shallow Cumulus Convection)

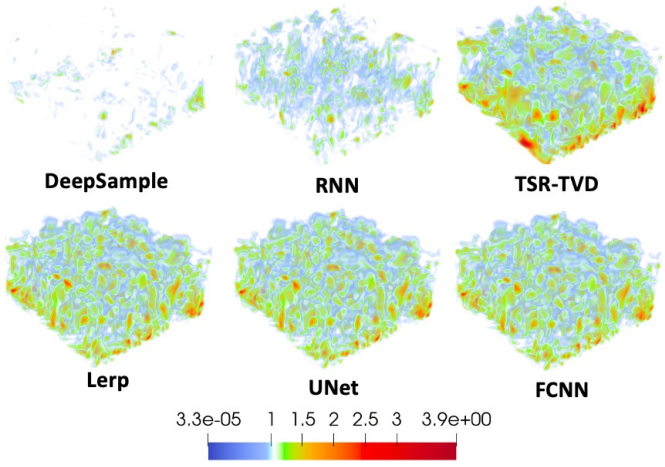


Figure 8. Visualization of RMSE (Shear Boundary Layer)

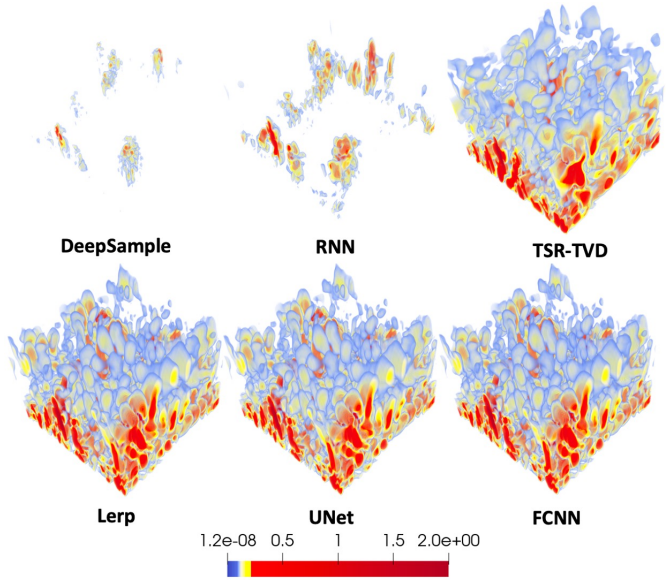


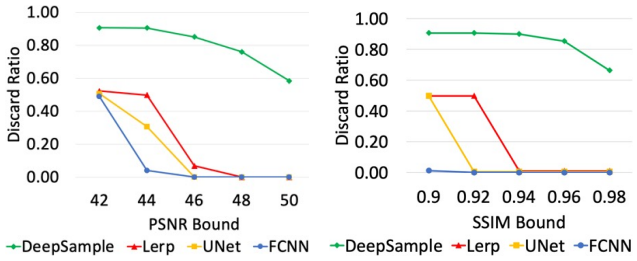
Figure 9. Visualization of RMSE (Shallow Cumulus Convection)

5.3.2 Q2: Discard Ratio

In the second set of experiments, we study the discard ratio of DeepSample under different quality bounds. In particular, we use Lerp, UNet, and FCNN to replace the MDDR module as the temporal data reconstruction function in our ECAS model, which allows them to make adaptive sampling decisions to meet the corresponding reconstruction quality bounds of the application for dynamic data. Please note that we do not include the RNN and TSR-TVD for this experiment because they can only work with the uniformly sampled dataset and are unable to handle data that are dynamically sampled. In particular, we set the quality bound using both PSNR (from 42 to 50) and SSIM (from 0.92 to 0.98), which ensures reasonable rendering effects of the reconstructed simulation data for effective post hoc analysis. The evaluation results are shown in Figure 10 and Figure 11. We observe that our DeepSample scheme significantly outperforms all compared baselines by achieving the highest discard ratio, which indicates the lowest I/O bandwidth and storage overhead. For example, our DeepSample achieves a performance gain of 27.2% compared to the best-performing baseline (i.e., Lerp) when we set the PSNR bound to be 42 on the Shallow Cumulus Convection dataset. Such performance gains demonstrate the effectiveness of the error-controlled deep data quality estimation design in the DeepSample that maximizes the discard ratio during the sampling process.

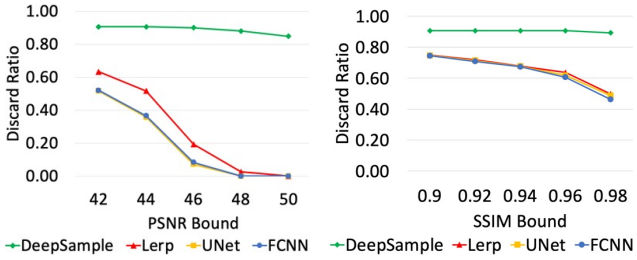
5.3.3 Q3: Meeting Quality Bounds

In the third set of experiments, we evaluate the performance of DeepSample in terms of meeting the quality bounds of the applications. The evaluation results are shown in Figure 12 and Figure 13. We observe that DeepSample consistently ensures the reconstructed data quality (i.e., Actual PSNR and Actual SSIM) is above the corresponding quality bounds of the application while keeping a high discard ratio. For example, DeepSample achieves a reconstructed data quality of 46.79 on PSNR (bound is 46) with a discard ratio



(a) PSNR Bound (b) SSIM Bound

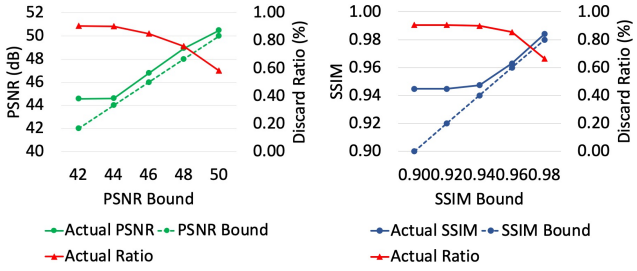
Figure 10. Performance of All Compared Schemes on Discard Ratios (Shear Boundary Layer)



(a) PSNR Bound (b) SSIM Bound

Figure 11. Performance of All Compared Schemes on Discard Ratios (Shallow Cumulus Convection)

of 85% on the Shear Boundary Layer dataset. The results indicate that DeepSample can reduce the I/O bandwidth by 85% while maintaining a high reconstructed data quality for post hoc analysis. In addition, we observed that the discard ratio of DeepSample decreases slightly when both PSNR and SSIM bounds increase. The reason is intuitive: DeepSample needs to sample more data (i.e., lower discard ratio) to meet higher quality bounds. Additionally, we present a fine-grained illustration of adaptive sampling decisions made by DeepSample over 50 timesteps for both datasets in Figure 14. We observe that DeepSample dynamically adjusts the sampling window to meet the specified quality bounds of the applications.

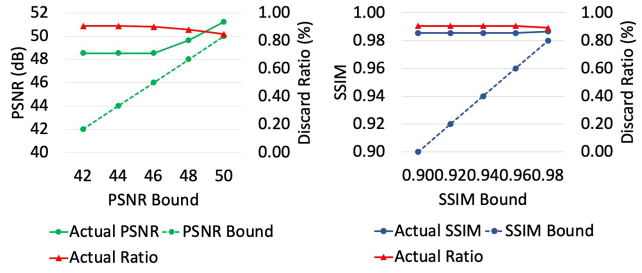


(a) PSNR Bound (b) SSIM Bound

Figure 12. Performance of DeepSample on Adaptive Temporal Sampling (Shear Boundary Layer)

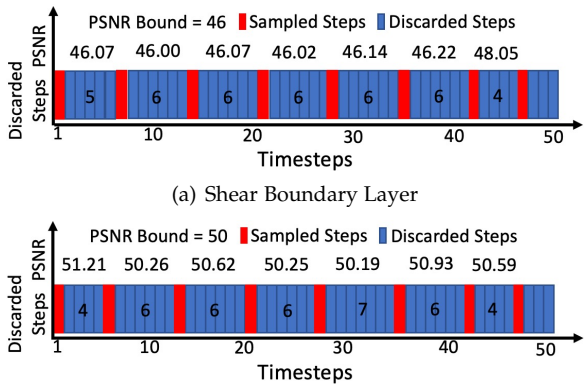
5.3.4 Q5: Improving the Performance of Lossy Compressor

In the fourth set of experiments, we show that DeepSample can also be used to help the spatial quality-aware data reduction methods (e.g., Lossy Compressors [14]) to improve their data compression ratio while maintaining the same level of data reconstruction quality. In particular, we



(a) PSNR Bound (b) SSIM Bound

Figure 13. Performance of DeepSample on Adaptive Temporal Sampling (Shallow Cumulus Convection)



(a) Shear Boundary Layer

(b) Shallow Cumulus Convection

Figure 14. Examples of Adaptive Temporal Sampling by DeepSample

compare the compression ratio of lossy compressor with and without DeepSample under the same quality bound. We use the mean absolute error (MAE) metric, which is the primary metric used to evaluate the performance of compressors. The evaluation results are shown in Table 1 and Table 2. We observe that DeepSample can significantly improve the compression ratio of lossy compressor while keeping the compression errors within bounds in all settings.

Table 1
Performance Comparisons on Compression Ratio With Lossy Compressor (Shear Boundary Layer)

Algorithm	MAE = 0.1	MAE = 0.05
Lossy Compressor + DeepSample	306.028	142.179
Lossy Compressor only	27.726	24.454

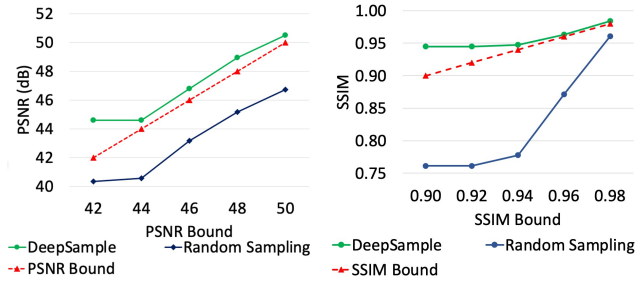
Table 2
Performance Comparisons on Compression Ratio With Lossy Compressor (Shallow Cumulus Convection)

Algorithm	MAE = 0.1	MAE = 0.05
Lossy Compressor + DeepSample	321.23	226.38
Lossy Compressor only	30.196	21.280

5.3.5 Q5: Ablation Study of DeepSample Scheme

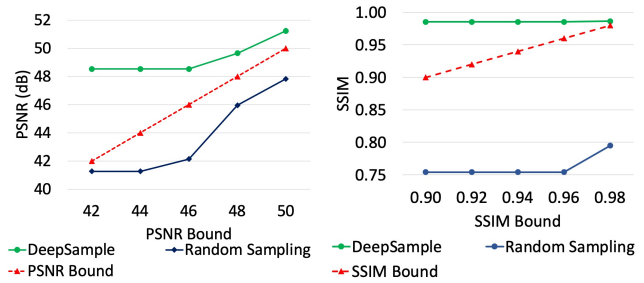
In the fifth set of experiments, we perform an ablation study to study the contribution of each component of DeepSample

to the reconstructed data quality. In particular, we first evaluate the effectiveness of our error-controlled adaptive sampling (ECAS) design by replacing it with a random sampling scheme. In particular, we randomly sample the same amount of selected data samples as our DeepSample scheme with varying sizes of sampling windows under each reconstruction quality bound and apply the same MDDR network to reconstruct the discarded data samples. The results are shown in Figure 15 and Figure 16. We observe that our ECAS design makes a clear contribution in improving the reconstructed data quality compared to the random sampling baseline while ensuring the reconstructed data quality straightly meets the corresponding quality bounds. We also evaluate the effectiveness of our multi-branch deep data reconstruction (MDDR) design by replacing it with a single branch reconstruction network design (SingleBranch) [22]. In particular, SingleBranch first leverages the extraction network to extract the deep features for the consecutively selected data samples. SingleBranch then applies the linear interpolation to interpolate the deep features for discarded data samples with varying sizes. Finally, SingleBranch leverages a single reconstruction network to reconstruct each discarded data sample using the interpolated deep features. The results are shown in Figure 17 and Figure 18. We observe that our DeepSample clearly improves the discarded ratios compared to SingleBranch under different reconstruction performance bound settings.



(a) PSNR Bound (b) SSIM Bound

Figure 15. Effectiveness of Error-Controlled Adaptive Sampling (Shear Boundary Layer)

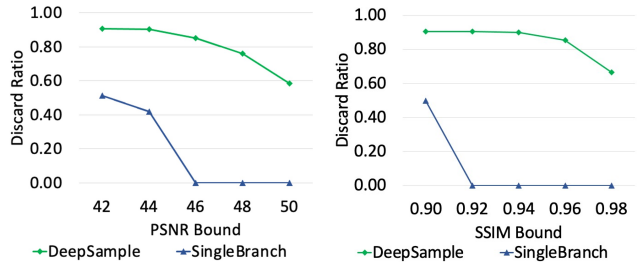


(a) PSNR Bound (b) SSIM Bound

Figure 16. Effectiveness of Error-Controlled Adaptive Sampling (Shallow Cumulus Convection)

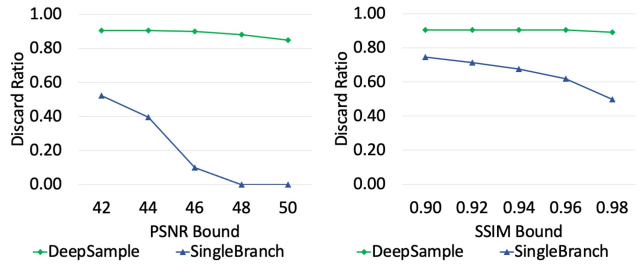
5.3.6 Q6: Parameter Analysis of DeepSample Scheme

In the last set of experiment, we study the performance of our DeepSample in terms of the discarded ratio by varying the number of branches (i.e., the number of reconstruction networks N) in our MDDR module. In particular, we evaluate the performance of our DeepSample by varying N from 2 to 10 under different PSNR and SSIM bounds [3]. The results are shown in Figure 19 and Figure 20. We observe that our DeepSample achieves a higher discarded ratio when N increases. This is because, with more reconstruction networks, our DeepSample is able to reconstruct the discarded data samples from a larger sampling window that meets the desirable reconstruction quality bound. Therefore, our DeepSample could effectively discard more data samples in the intermediate steps and achieve a better discarded ratio. In addition, we also observe that our DeepSample can achieve a reasonable discard ratio when N is relatively small (e.g., our scheme can achieve above 80% discarded ratios when N equals 6 in most cases).



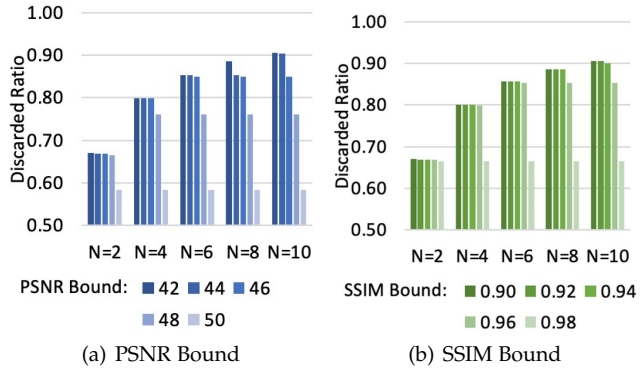
(a) PSNR Bound (b) SSIM Bound

Figure 17. Effectiveness of Multi-Branch Deep Data Reconstruction (Shear Boundary Layer)



(a) PSNR Bound (b) SSIM Bound

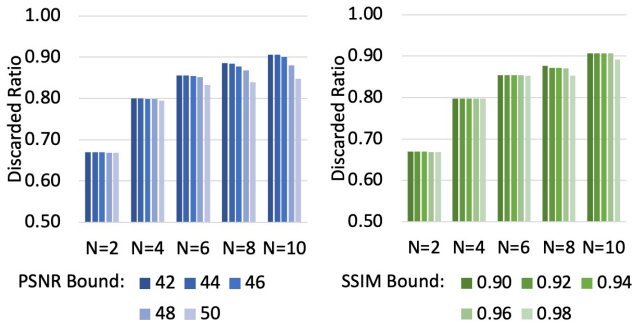
Figure 18. Effectiveness of Multi-Branch Deep Data Reconstruction (Shallow Cumulus Convection)



(a) PSNR Bound (b) SSIM Bound

Figure 19. Parameter Analysis of DeepSample Scheme on Varying Number of Reconstruction Network (Shear Boundary Layer)

3. Note that we stop at $N = 10$ due to the memory space constraint on our GPU server.



(a) PSNR Bound (b) SSIM Bound

Figure 20. Parameter Analysis of DeepSample Scheme on Varying Number of Reconstruction Network (Shallow Cumulus Convection)

6 Conclusion

This paper presents the DeepSample framework to address an in situ adaptive data sampling and reconstruction problem in scientific simulation applications. DeepSample addresses two challenges: in situ adaptive sampling for complex scientific simulation data and error-controlled non-uniform data reconstruction. We develop a multi-branch deep decoder network based approach to effectively reconstruct the discarded samples with rigorous quality assurance. The results on two real-world scientific simulations show that DeepSample significantly outperforms other representative methods on both data discard ratios and reconstructed simulation data quality. We believe DeepSample will provide useful insights to address similar big data challenges motivated by the high-velocity and large-volume nature of the data beyond scientific simulations.

Acknowledgment

This research is supported in part by the National Science Foundation under Grant No. IIS-2008228, CNS-1845639, CNS-1831669, Army Research Office under Grant W911NF-17-1-0409. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] J. Wang, S. Hazarika, C. Li, and H.-W. Shen, "Visualization and visual analysis of ensemble data: A survey," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 9, pp. 2853–2872, 2018.

[2] H. Mizuta and Y. Yamagata, "Agent-based simulation and greenhouse gas emissions trading," in *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304)*, vol. 1. IEEE, 2001, pp. 535–540.

[3] S. Hackstein, F. Vazza, M. Brüggen, J. G. Sorce, and S. Gottlöber, "Simulations of ultra-high energy cosmic rays in the local universe and the origin of cosmic magnetic fields," *Monthly Notices of the Royal Astronomical Society*, vol. 475, no. 2, pp. 2519–2529, 2018.

[4] P. D. Morris, A. Narracott, H. von Tengg-Kobligk, D. A. S. Soto, S. Hsiao, A. Lungu, P. Evans, N. W. Bressloff, P. V. Lawford, D. R. Hose *et al.*, "Computational fluid dynamics modelling in cardiovascular medicine," *Heart*, vol. 102, no. 1, pp. 18–28, 2016.

[5] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE, 2016, pp. 730–739.

[6] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. Nashed, and T. Peterka, "Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 23–33, 2019.

[7] I. Foster, M. Ainsworth, B. Allen, J. Bessac, F. Cappello, J. Y. Choi, E. Constantinescu, P. E. Davis, S. Di, W. Di *et al.*, "Computing just what you need: Online data analysis and reduction at extreme scales," in *European conference on parallel processing*. Springer, 2017, pp. 3–19.

[8] T. Peterka, D. Bard, J. Bennett, E. W. Bethel, R. Oldfield, L. Pouchard, C. Sweeney, and M. Wolf, "Ascr workshop on in situ data management: Enabling scientific discovery from diverse data sources," 2 2019.

[9] K.-L. Ma, "In situ visualization at extreme scale: Challenges and opportunities," *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 14–19, 2009.

[10] J. Woodring, J. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann, "In-situ sampling of a large-scale particle simulation for interactive visualization and analysis," in *Computer Graphics Forum*, vol. 30, no. 3. Wiley Online Library, 2011, pp. 1151–1160.

[11] A. Biswas, S. Dutta, J. Pulido, and J. Ahrens, "In situ data-driven adaptive sampling for large-scale simulation data summarization," in *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, 2018, pp. 13–18.

[12] Y. Su, G. Agrawal, J. Woodring, K. Myers, J. Wendelberger, and J. Ahrens, "Taming massive distributed datasets: data sampling using bitmap indices," in *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, 2013, pp. 13–24.

[13] J. Han and C. Wang, "Tsr-tvd: Temporal super-resolution for time-varying data analysis and visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 205–215, 2019.

[14] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 438–447.

[15] X. Tong, T.-Y. Lee, and H.-W. Shen, "Salient time steps selection from large scale time-varying data sets with dynamic time warping," in *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 2012, pp. 49–56.

[16] B. Zhou and Y.-J. Chiang, "Key time steps selection for large-scale time-varying volume datasets using an information-theoretic storyboard," in *Computer Graphics Forum*, vol. 37, no. 3. Wiley Online Library, 2018, pp. 37–49.

[17] J. Zhou, O. C. Au, G. Zhai, Y. Y. Tang, and X. Liu, "Scalable compression of stream cipher encrypted images through context-adaptive sampling," *IEEE transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1857–1868, 2014.

[18] H. Luo, J. Wang, Y. Sun, H. Ma, and X.-Y. Li, "Adaptive sampling and diversity reception in multi-hop wireless audio sensor networks," in *2010 IEEE 30th International Conference on Distributed Computing Systems*. IEEE, 2010, pp. 378–387.

[19] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017, pp. 1129–1139.

[20] Z. Zhou, Y. Hou, Q. Wang, G. Chen, J. Lu, Y. Tao, and H. Lin, "Volume upscaling with convolutional neural networks," in *Proceedings of the Computer Graphics International Conference*, 2017, pp. 1–6.

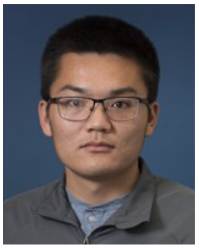
[21] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.

[22] A.-D. Nguyen, W. Kim, J. Kim, and S. Lee, "Video frame interpolation by plug-and-play deep locally linear embedding," *arXiv preprint arXiv:1807.01462*, 2018.

[23] A. Shoshani and D. Rotem, *Scientific data management: challenges, technology, and deployment*. CRC Press, 2009.

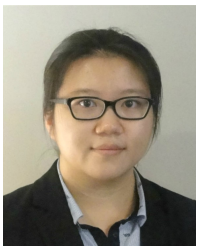
[24] W. F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck *et al.*, "Adios 2: The adaptable input output system. a framework for high-

performance data management," *SoftwareX*, vol. 12, p. 100561, 2020.
[25] P. G. Lindstrom *et al.*, "Fpzip," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2017.
[26] H. Obermaier and K. I. Joy, "Future challenges for ensemble visualization," *IEEE Computer Graphics and Applications*, vol. 34, no. 3, pp. 8–11, 2014.
[27] J. Han, J. Tao, and C. Wang, "Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces," *IEEE transactions on visualization and computer graphics*, 2018.
[28] F. Hong, J. Zhang, and X. Yuan, "Access pattern learning with long short-term memory for parallel particle tracing," in *2018 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 2018, pp. 76–85.
[29] B. Kim and T. Günther, "Robust reference frame extraction from unsteady 2d vector fields with convolutional neural networks," in *Computer Graphics Forum*, vol. 38, no. 3. Wiley Online Library, 2019, pp. 285–295.
[30] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.
[31] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4491–4500.
[32] J. Zhang, S. H. Yeung, Y. Shu, B. He, and W. Wang, "Efficient memory management for gpu-based deep learning systems," *arXiv preprint arXiv:1903.06631*, 2019.
[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[34] Y. Li, A. Mahjoubfar, C. L. Chen, K. R. Niazi, L. Pei, and B. Jalali, "Deep cytometry: deep learning with real-time inference in cell sorting and flow cytometry," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.
[35] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.
[36] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.
[37] C.-H. Moeng and P. P. Sullivan, "A comparison of shear-and buoyancy-driven planetary boundary layer flows," *Journal of the Atmospheric Sciences*, vol. 51, no. 7, pp. 999–1022, 1994.
[38] A. P. Siebesma, C. S. Bretherton, A. Brown, A. Chlond, J. Cuxart, P. G. Duynkerke, H. Jiang, M. Khairoutdinov, D. Lewellen, C.-H. Moeng *et al.*, "A large eddy simulation intercomparison study of shallow cumulus convection," *Journal of the Atmospheric Sciences*, vol. 60, no. 10, pp. 1201–1219, 2003.
[39] N. H. de Hoon, A. C. Jalba, E. Eisemann, and A. Vilanova, "Temporal interpolation of 4d pc-mri blood-flow measurements using bidirectional physics-based fluid simulation." in *VCBM*, 2016, pp. 59–68.
[40] M. Livne, J. Rieger, O. U. Aydin, A. A. Taha, E. M. Akay, T. Kossen, J. Sobesky, J. D. Kelleher, K. Hildebrand, D. Frey *et al.*, "A u-net deep learning framework for high performance vessel segmentation in patients with cerebrovascular disease," *Frontiers in neuroscience*, vol. 13, p. 97, 2019.
[41] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
[42] S. Wiewel, M. Becher, and N. Thuerey, "Latent space physics: Towards learning the temporal evolution of fluid flow," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 71–82.

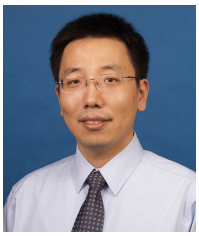


Yang Zhang is a PhD student in the Department of Computer Science and Engineering at the University of Notre Dame. He received his M.S. degree from Indiana University-Bloomington in 2017 and a B.S. degree from Wuhan University in 2013. His research interests include social sensing, machine learning, deep learning, and cyber-physical systems. He is a student member of IEEE.

Hanqi Guo is an assistant computer scientist at Argonne National Laboratory, scientist at the University of Chicago Consortium for Advanced Science and Engineering (CASE), and fellow of the Northwestern Argonne Institute for Science and Engineering (NAISE). His research interests include data analysis, visualization, and machine learning for scientific data. He has published more than 40 research papers in top visualization journals and conferences including IEEE VIS, IEEE TVCG, and IEEE TPDS. He is also the recipient of the best paper award in IEEE VIS 2019 and the winner of the 2017 Postdoctoral Performance Award in Basic Research in Argonne National Laboratory. He received his Ph.D. degree in computer science from Peking University in 2014 and his B.S. degree in mathematics and applied mathematics from Beijing University of Posts and Telecommunications in 2009.



Lanyu Shang is a Ph.D. student in the School of Information Sciences at the University of Illinois Urbana-Champaign. She received an M.S. in Data Science from New York University and a B.S. in Applied Mathematics from the University of California - Los Angeles (UCLA). Her research interest primarily lies in online misinformation detection using social media data. She is a student member of IEEE.



Dong Wang received his Ph.D. in Computer Science from University of Illinois Urbana-Champaign (UIUC) in 2012. He is now an associate professor in the School of Information Sciences at the University of Illinois Urbana-Champaign. Dr. Wang's research interests lie in the area of reliable social sensing, human-centric AI, cyber-physical computing, and smart city applications. He received the NSF CAREER award in 2019, Google Faculty Research Award in 2018, Army Research Office Young Investigator Program (YIP) Award in 2017, Wing-Kai Cheng Fellowship from the University of Illinois in 2012 and the Best Paper Award of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2010. He is a member of IEEE and ACM.

Tom Peterka is a computer scientist at Argonne National Laboratory, a scientist at the University of Chicago Consortium for Advanced Science and Engineering (CASE), an adjunct assistant professor at the University of Illinois at Chicago, and a fellow of the Northwestern Argonne Institute for Science and Engineering (NAISE). His research interests are in large-scale parallel in situ analysis of scientific data. Recipient of the 2017 DOE Early Career Award and four best paper awards, Peterka has published over 100 peer-reviewed papers in conferences and journals that include ACM/IEEE SC, IEEE IPDPS, IEEE VIS, IEEE TVCG, and ACM SIGGRAPH. Peterka received his Ph.D. in computer science from the University of Illinois at Chicago in 2007, and he currently leads several DOE- and NSF-funded projects.