


# XEventNet: Extreme Weather Event Prediction using Convolutional Neural Networks and In Situ Visualization

Muzafar Ahmad Wani and Preeti Malakar 

Indian Institute of Technology Kanpur, India

## Abstract

Extreme weather phenomena such as cyclones, torrential rainfall, snow storms, flash floods and landslides pose serious threat to living beings and property all over the world. An accurate and early prediction system for these extreme events may minimize the loss of life and property. However, this requires an online prediction system integrated with the weather simulation model for faster prediction such that low I/O bandwidth does not hinder performance. We present an in situ framework, XEventNet, that integrates weather simulation, deep learning-based prediction, and visualization. XEventNet predicts extreme events at real-time while the simulation is running using a Convolutional Neural Network (CNN). XEventNet is trained and tested on 400 events (extreme and non-extreme). Data is streamed online from XEventNet simulation processes to prediction processes for parallel inference. XEventNet uses the prediction values with high confidence to selectively transfer sub-domains of the large parent simulation domain. We use ADIOS2 for parallel data transfers via memory between groups of processes. This helps in timely prediction and visualization of critical weather events despite large volume of simulation data. We performed weather simulations at 9 km resolutions, thereby producing gigabytes of data per time step. XEventNet is able to classify four extreme events at real-time and visualize the same. We achieved an average prediction accuracy of 90.25% for all extreme events using a single CNN model. We ran weather simulations on up to 512 processes and parallel predictions on up to 64 processes, thereby streaming gigabytes of data in parallel within seconds. This was possible due to efficient data transfer and process mapping. Furthermore, our selective data transfer for visualization resulted in more than 70% reduction in data size, thereby improving the end-to-end simulation-prediction-visualization times.

Visualization application domain Scientific Visualization

## CCS Concepts

• **Human-centered computing** → Visualization; • **Computing methodologies** → Parallel computing methodologies;

## 1. Introduction

Extreme weather events such as hurricanes, cyclones, flash floods, heavy rains, severe storms and landslides are common phenomena across the globe. It is challenging to numerically model these phenomena due to the complex and sometimes unknown interactions of several meteorological variables. The weather forecast models are continuously being improved to accurately predict and with higher lead times. Extreme events are likely to occur more frequently in the near future due to climate change [Sto16]. Worldwide data estimates occurrence of about 200 – 16K fatalities per year globally [Ser24, UCL24]. It is reported that more than 2000 fatalities per year occur in the Indian subcontinent due to extreme events [IMD24, Ind15]. Thus extreme weather forecasting and monitoring is an important problem.

Machine learning (ML) and recently deep learning (DL) is being increasingly used by climate and weather scientists to predict weather and to predict occurrence of extreme events [Sto16, KSH\*23, BXZ\*23, SEH\*20, QZZ\*17, CHG\*23, TSP\*24]. However, prior

works using ML/DL models focus on predicting one extreme event primarily. Various countries and regions across the globe such as the United States of America and Southeast Asia witness a diverse spectrum of extreme events with rising frequency and severity in recent years. These events include long intense rainfall, devastating floods, severe heat waves, cyclones, hurricanes, forest fires, and landslides. Thus it is useful to have a comprehensive prediction model for all events. Most prior works using ML/DL models to identify extreme events are not deployed online to analyze an ongoing simulation. There are challenges to stream data from an ongoing simulation to perform inference. This is because the simulation and inference are fundamentally different codes with different scaling characteristics and performance attributes. There are also synchronization challenges that needs to be accounted for so that the compute-intensive simulation application does not stall for longer durations.

Weather forecast models output a large volume of data (order of terabytes) for high-resolution high-fidelity simulations. This data is usually stored and analyzed to derive insights from the simulations.

Visualization of such large volume of data is necessary for quick comprehension of output. Typically, the output frequency and hence the analysis and visualization frequency is low due to the cost involved in data transfer from simulations to visualization. However, it is imperative to visualize critical events as early as possible so that appropriate actions may be taken by policy makers in time. Thus, a predictive comprehensive system for real-time extreme weather event prediction and visualization may be helpful.

We have developed XEventNet (eXtreme Event prediction using deep neural Networks) for in situ predictive analysis of extreme weather events and in situ visualization of the same. We have used convolutional neural network (CNN) for predictions. We used a CNN model with separate branches for 2D and 3D data that captures the features separately for 2D and 3D meteorological variables. We used distributed TensorFlow [QHDT20, CL22] for efficient prediction at runtime. We trained the model on four major extreme events – tropical cyclones, heavy rainfall, landslides and snowfall. XEventNet may serve as an early warning system to prepare for and mitigate the impact of severe weather events. XEventNet combines ongoing weather simulation, and online prediction model and an online/in situ visualization system. Here in situ refers to concurrent analysis and visualization as described in [Cea20]. We use parallel weather simulations, parallel prediction models and parallel visualization for efficiency. XEventNet uses the high performance I/O library ADIOS [Gea20] for in situ data transfer between the parallel simulation and parallel prediction model, as well as between parallel prediction model to parallel visualization application. This avoids the I/O costs and significantly reduces the end-to-end latency. The entire domain of large simulation output are often not required to be transferred due to non-significant changes across timesteps. Thus based on the predictions, we also propose a low-latency data reduction mechanism to reduce the data volume for data transfer, this further improves the efficiency of XEventNet. We used CPUs for all computations in this work. We plan to use multiple GPUs for parallel inference in future.

XEventNet continuously predicts and visualizes extreme events from simulation output in real-time as the simulation progresses without writing to disk. We used the popular operational weather forecasting model WRF (Weather Research and Forecasting) [SKD\*08] for weather simulation of Asian region where multiple extreme events occur every year. We demonstrate results from extreme events of more than 15 years in this region. The main contributions of this work are (1) An end-to-end integrated in situ workflow for extreme weather event predictions at real-time with coupled simulation, inference and visualization (all parallel). (2) A parallel multi-class CNN model to identify four distinct extreme events with more than 85% accuracy with separate branches for 2D and 3D meteorological variables to improve accuracy. (3) A fully in-memory pipeline to seamlessly perform asynchronous data transfers between simulations and inference processes and then to visualization processes eliminating I/O overhead. (4) Selective data streaming based on CNN confidence scores, reducing data transfer overhead by 70% to improve the end-to-end scalability. XEventNet CNN model was trained and tested using a dataset of 500 extreme events. We generated this dataset using WRF simulation. We achieved an average accuracy of 90.1% in predicting extreme events. We also achieved an average 74.28% reduction in data transfer time for vi-

ualization. We used distributed TensorFlow on a maximum of 64 CPU cores where data was streamed using ADIOS from a maximum of 512 processes running the WRF simulation. This delivered a low online inference time of 5.92 seconds. The average end-to-end simulation time was 5.03 minutes for one day of simulation, out of which the simulation executes for 4.8 minutes and our approach adds an overhead of merely 0.23 minutes for online prediction and visualization. The idle time is low due to pipelining.

The rest of the paper is organized as below. Section 2 surveys related work as well as describes some background on ADIOS. Section 3 describes our XEventNet framework in detail. Section 4 describes the implementation details of XEventNet. Section 5 describes our evaluation setup, dataset and demonstrates the utility of XEventNet through various results. Section 6 concludes our work and mentions future directions.

## 2. Related Work and Background

We first survey existing work on deep learning for weather prediction and in situ visualization. Then we provide an overview of ADIOS [Gea20] which we used to stream data from simulation processes to visualization processes (memory to memory data transfer).

### 2.1. Related Work

#### 2.1.1. Deep Learning for Weather Prediction

Machine learning (ML) and deep learning (DL) models have been used for weather prediction in several studies [KSH\*23, BXZ\*23, SEH\*20, NBK\*23, CHG\*23]. Convolutional Neural Networks (CNN) [QZZ\*17, WT22], Transformers [CHG\*23, ZCF\*24], Long Short-Term Memory (LSTM) [FISA22] and hybrid models [SCW\*15] have been widely explored for weather forecasting. Fan et al. [FISA22] presented weather forecasting using a CNN-LSTM model. It predicts multiple variables such as cloud mask, temperature, and rainfall with low mean squared error values for all variables. FourCastNet [KSH\*23] utilizes Adaptive Fourier Neural Operator (AFNO) for high-resolution weather modeling using deep learning. PanGu [BXZ\*23] uses a 3D neural network and has higher accuracy than FourCastNet. MetNet [SEH\*20] uses a combination of CNNs, LSTMs, and auto-encoders for forecasting. Quie et al. [QZZ\*17] used a multi-task convolutional neural network to learn from multiple sites to predict one event. These existing models [SEH\*20, KSH\*23, BXZ\*23, BXZ\*22] predict weather and extreme events and are computationally more expensive than XEventNet due to I/O. They also do not integrate data streaming over memory from multiple parallel simulation processes to multiple inference processes. The models also do not include real-time visualization of extreme events which may be helpful for policy makers. In contrast, XEventNet predicts four distinct events at real-time through continuous parallel streaming of sub-domains and parallel real-time inference and visualization.

ML and DL models have also been used to predict extreme events. Trivedi et al. [TSP\*24] used Multi-layer perception (MLP) and CNN separately for rainfall prediction using WRF output. Matsuoka et al. [MNSU18] developed a 2D CNN model using simulated 20 year data of outgoing long wave radiation (OLR) for detecting

precursors and tropical cyclones (TC). Mukherjee et al. [MM22] used a number of variables from WRF simulated output to predict cyclogenesis with a high accuracy. Liu et al. [Lea16] presented deep CNN techniques to detect extreme climate events such as tropical cyclones and weather fronts with high accuracy. Khaira et al. [KCTA24] used WRF simulations output data with ML models to enhance snowfall predictions with an accuracy up to 90%. All above works predict weather events *post hoc*, while we present an online/*in situ* framework for extreme weather event prediction. Further, XEventNet uses a single model that is trained to predict four diverse extreme events online and efficiently by avoiding I/O cost. Further we used distributed TensorFlow for parallel inference using data streamed from the simulations.

### 2.1.2. Data Streaming for In Situ Visualization

Simulations generate massive amount of data in a few hours. Several libraries exist to analyze and visualize large data, such as DataSpaces [DPK10], Flexpath [DBE\*14], and Decaf [DP17]. VisIt Libsim [WFM11a], Paraview Catalyst [ABG\*15], Ascent [Aea25] and others are used for in situ visualization. ADIOS [Gea20] has been used in prior works [LF22, FDJ\*23] for efficient data movement between producer and consumer. There exists work on data reduction [LJ14, PVH\*03] using compression and by extracting features from simulation data. In contrast to these works, XEventNet performs in situ analysis of extreme events followed by in situ visualization. We used VisIt LibSim for in situ visualization and ADIOS for data streaming. We provide a brief overview of ADIOS.

## 2.2. Background: Data Streaming using ADIOS

Adaptable Input Output (I/O) System (ADIOS) [Gea20, Eea24] is a high-performance data management framework designed for efficient low-latency data transfer in scientific computing environments. It is one of the high-performance I/O libraries that provides flexible data organization. ADIOS provides various options to transfer large volumes of data generated by simulations based on file-based I/O and data staging methods. We used one of the data staging methods that focus on memory-to-memory communication and enable in situ workflows. Here data is consumed and processed without being written to persistent storage. Examples include SST (Staging Streaming Transport), DataMan, InSituMPI and Single Sided Communication. We used the ADIOS2 SST (Sustainable Staging Transport) engine for efficient data transfer. Data streamed by ADIOS2 can be directly used in situ visualization and not written to secondary storage. The write/read APIs of ADIOS are used for direct data streaming from producers to consumers.

## 3. XEventNet: Extreme Event Prediction

Real-time prediction, analysis, and visualization of extreme weather events such as cyclones, hurricanes, heavy rainfall, snowstorms, landslides, and flash floods may help reduce the devastation caused by such events every year across the globe. Therefore, it is necessary to analyze and predict the output of numerical mesoscale weather models as soon as possible. We have developed a real-time deep learning-based prediction model, XEventNet, that predicts extreme events such as cyclones, heavy rainfall, snowfall and landslides and

performs in situ visualization of these events. In this section, we first explain the overall workflow of our model including in situ prediction, data streaming, and in situ visualization. Next, we explain our deep learning (DL) prediction model, followed by parallelization of DL model to improve efficiency.

### 3.1. In Situ Prediction Workflow

XEventNet directly processes the data (multi-variable) generated by the simulations in memory every few time steps. This frequency may be set by the user. A high temporal frequency may be used for such analysis because the output data is directly transferred from simulation processes to analysis/prediction processes (process-to-process transfer) without involving the disks for I/O. The output data is intercepted by our library for data transfer via asynchronous interprocess communication. Figure 1 shows the entire workflow of simulation, prediction and visualization components. The leftmost rectangle shows 8 simulation processes, i.e. the simulation domain is divided into 8 sub-domains.

The simulation processes (writers) concurrently stream data (sub-domains) to the prediction processes (readers) via ADIOS SST (Sustainable Staging Transport) as explained in Section 2.2. The ratio  $M:N$  of the number of simulation and prediction processes is tunable. However, we assume that  $M$  will always be greater than  $N$  because simulation is more compute intensive. Thus several simulation processes may stream data to 1 prediction process. This ratio is 2 in the figure, there are 4 prediction processes. XEventNet prediction processes computes the offset of data in the readers/consumers according to the ratio  $M:N$  which is known apriori. Note there is no storage involved in this data transfer and multiple transfers occur simultaneously. This improves the efficiency of XEventNet.

Every sub-domain is divided into batches of  $10 \times 10$  grids in the prediction processes for parallelization.  $10 \times 10$  grids represent a physical domain of  $90 \text{ km} \times 90 \text{ km}$  which is larger than the extents of the extreme events considered. The average extent of the four events is  $1500 \text{ sq. km}$ . which is well within our grid size. For example, the eye of a cyclone extends from  $10 - 50 \text{ km}$ , landslides extend to a maximum of  $1 \text{ square km}$  area [IMD24]. Larger grid sizes may imply coarser granularity for an event, thus it impacts accuracy and it limits parallelizability. Smaller grid sizes imply more processing time for CNN. Each prediction process then runs the trained CNN model (described next) to infer the presence of extreme events. This is done in parallel on each prediction process. The output of the CNN model is analyzed to determine the importance of the sub-domains. The prediction process with at least one positive prediction for an extreme event sends its data for visualization. Every simulation time step may generate gigabytes of data (2D and 3D meteorological variables) depending on the simulation resolution. We do not send the entire domain for visualization for better end-to-end throughput. There may be none or few sub-domains sent for visualization depending on the output of CNN model. An example of two such sub-domains is shown in Figure 1.

### 3.2. CNN Model to Identify Extreme Events

Our proposed architecture for XEventNet uses a convolution neural network (CNN) to process both 2D and 3D simulation variables, as

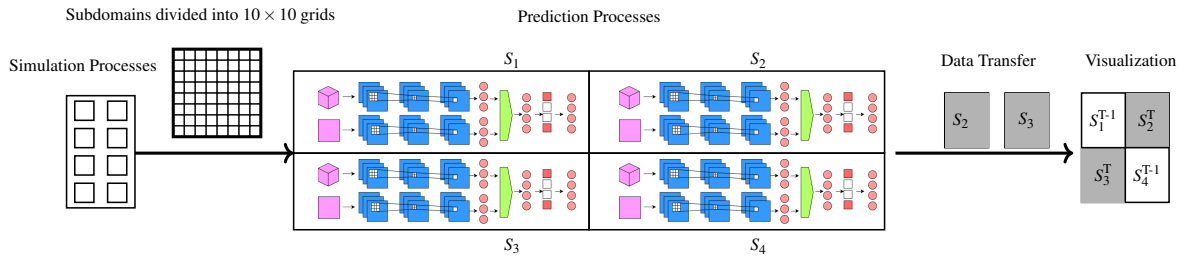


Figure 1: XEventNet: In Situ Extreme Weather Event Prediction Framework. Parallel streaming from multiple simulation processes to multiple prediction processes, parallel data streaming of subset of domains from prediction processes to visualization processes.

shown in Figure 2. CNN model is a type of deep learning model that uses convolution layers to automatically find important features in the data. CNN uses a kernel (small matrix) that moves across the input grid and performs element-wise aggregations to extract useful features. We developed a new model instead of using existing models [SEH\*20, KSH\*23, BXZ\*22] because these predict offline one kind of extreme event. They incur higher prediction times because they do not perform prediction at real-time from memory. XEventNet also streamlines parallel inference by streaming data from multiple simulation sub-domains concurrently. XEventNet CNN model is trained using data points corresponding to extreme events and also data points that do not contain any extreme event. Each labeled data point corresponds to a specific geographic location. This information is recorded in a separate location file; it includes the latitude and longitudinal coordinates and the extreme event label (cyclone, heavy rainfall, landslide, snowfall or none). For each data point, XEventNet extracts a  $10 \times 10$  small square centered at the location (latitude, longitude) from the full domain data for each of the 20 variables of interest (Table 1). This square is aligned with WRF's grid cells, each of which are  $9 \text{ km} \times 9 \text{ km}$  in the physical domain. The spatial extent (rather than a single data point) is relevant in weather simulations due to the nature of the numerical simulations.

The 2D variables are arranged in a  $10 \times 10 \times 12$  grid, where 12 represents the number of 2D variables. The 3D variables are arranged in a  $10 \times 10 \times Z \times 8$  grid, where  $Z$  is the depth of the 3D variables (typically 27) and 8 is the number of 3D variables. The 2D and 3D grids are passed to the model for training. Our model consists of two distinct branches that handle 2D and 3D data independently as shown in Figure 2. The two separate branches are concatenated after a few layers. We used two branches to avoid losing important information from the 3D variables [FYS\*19]. The model is able to fully capture the depth information of the 3D variables.

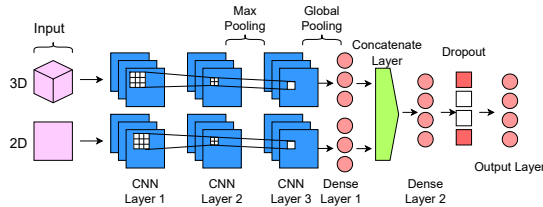


Figure 2: XEventNet Model: CNN Model for Extreme Event Prediction with separate branches for 2D and 3D Variables.

We use three Conv2D/Conv3D layers for both branches. The first

layer has 32 filters and a kernel size of  $3 \times 3$  for 2D and  $3 \times 3 \times 3$  for 3D [AM20, GJGP22] capturing basic spatial features from the input. The output is a tensor of shape  $(H, W, 32)$ . The second layer has 64 filters and with the same kernel size as the first layer, enabling the extraction of more detailed and complex spatial patterns. The output is a tensor of shape  $(H, W, 64)$ . The third layer has 128 filters and uses the same kernel size. The output is  $(H, W, 128)$ . Each CNN layer is followed by a pooling operation to downsample the feature maps as shown in Figure 2. We employ a max-pooling layer with a pool size of  $2 \times 2$  for 2D and  $2 \times 2 \times 2$  for 3D data after the first and second layers. We employ a global max-pooling layer after the third CNN layer. A GlobalMaxPooling2D layer reduces each feature map to a single value, producing a vector of length 128, which is then passed through a Dense layer with 128 nodes and ReLU activation for further abstraction. These pooling layers help reduce the spatial dimensions while retaining key features. The 3D branch follows a similar structure, with three Conv3D layers of 32, 64, and 128 filters, same kernel size, ReLU activation, same padding, capturing spatiotemporal patterns in the 3D input. The output after the third Conv3D layer is a tensor of shape  $(D, H, W, 128)$ , which is then passed through a GlobalAveragePooling3D layer, producing a vector of length 128. This vector is further processed through a Dense layer with 128 nodes and ReLU activation.

We concatenate the output from the 2D and 3D branches into a single feature set. The combined features are passed to a fully connected dense layer, where the model further processes these features to learn useful patterns for predicting the final output. We added a dropout layer with a dropout rate of 0.5 to reduce overfitting by randomly dropping connections during training. This ensures that the model generalizes better for unseen data. The final classification is performed using a Dense output layer with 5 nodes and softmax activation, predicting probabilities across five classes (four extreme events and one class for no extreme event). We trained the model using the Adam optimizer with a learning rate of 0.005, which provided a balance between convergence speed and stability. The model was optimized using sparse categorical cross-entropy loss, ensuring efficient multi-class classification. The four classes are cyclone, heavy rainfall, landslide and snowfall. The large scale of distributed data required us to distribute the training as well. We discuss our parallelization approaches in the next section.

### 3.3. Parallel CNN Model

We used the popular TensorFlow [RGGGM19] as our deep learning framework. We used data-parallel deep learning using the parallel

TensorFlow on CPUs for both training and inference. We trained using weather simulation data on multiple MPI processes for reducing training times. We ran the trained model in parallel for inference at real-time as shown in Figure 1. In this example, the model runs in parallel on the 4 prediction processes. XEventNet divides the labeled input data points equally among the MPI processes. For each data point assigned to a process, the framework fetches a  $10 \times 10$  grid representing a spatial region of interest. Then the data is organized into batches by TensorFlow. We used batching [Lea19] for both training and inference to allow simultaneous processing of multiple input via tensors at once. This reduces the overall training and inference time because each process handles a large volume of data at once. TensorFlow ensures that these batches are processed in parallel. We found that the optimal batch size varies with the number of processes used, as demonstrated in Section 5.2.3.

Each process uses TensorFlow's parallel training capabilities to independently train on its assigned labeled data point subset. This ensures that each process maintains its own mirrored copy of the model, facilitating synchronous updates after each training iteration. After each training step, gradients from every process are synchronized, where the gradients are averaged and applied collectively. This ensures consistency, reinforcing the collective learning of the network. This distributed training approach accelerates processing, especially for large datasets such as weather simulation output. During inference, once all processes complete predictions for their batches, the results are gathered at a central process. Combined prediction results are then used for visualization in XEventNet.

### 3.4. Efficient Visualization of Extreme Weather Events

XEventNet performs visualization of the time steps that have probability of extreme events so that it is easy to comprehend severe events. There is a default output frequency of the simulation that also determines the prediction frequency and visualization frequency. XEventNet uses a high frequency because our predictions are fast due to parallelization. The data transfer time is small because we adopted a data reduction technique to reduce the volume of data that will be transferred from the prediction processes to the visualization processes using ADIOS SST. XEventNet also uses parallel visualization for fast rendering. XEventNet selectively transfers the most relevant sub-domains instead of transferring the entire simulation data per time step which may be in gigabytes. We exemplify this in Figure 1. In this example, only two out of the four sub-domains (greyed) are transferred from the prediction processes (ADIOS writers) to the visualization processes (ADIOS readers). The number of sub-domains that are eventually transferred depends on the predictions.

The sub-domain data received at the visualization processes at a time step  $T$  replaces the previously received time step data (greyed rectangles  $S_2^T$  and  $S_3^T$ ). The previously received sub-domain data is retained for the sub-domains that were predicted to not contain extreme events (white rectangles  $S_1^{T-1}$  and  $S_4^{T-1}$ ), as shown in the rightmost rectangle. The four rectangles indicate 4 visualization processes. Sub-domain mapping between inference and visualization processes is fixed for a configuration. This simplifies the overlay logic for visualization. The overlaying or patching of new and old sub-domains are done by the visualization processes

(ADIOS readers). This forms the full time step for in situ visualization. Relevant variables are visualized based on the type of extreme event. Thus XEventNet is able to visualize most critical weather events efficiently with minimum overhead. We integrated VisIt's LibSim [WFM11b] in situ library in XEventNet for visualization of data transferred from prediction processes to visualization processes. We integrated Libsim library functions in our reader code on the visualization processes for in situ visualization of variables of interest. We next describe some of our implementation details.

## 4. XEventNet Implementation Details

Figure 1 shows the overall components and the workflow of XEventNet. In this section, we describe the implementation details of XEventNet - simulation, prediction, visualization and data transfer between these components.

### 4.1. Weather Simulation

We used Weather Research and Forecasting (WRF) [SKD\*08] numerical weather simulation model in our work. WRF model outputs 156 variables. XEventNet uses 20 variables for prediction as shown in Table 1. We used 8 3D and 12 2D variables. We selected these variables from literature [Gra79, MM22, AF18, FGVS\*15, Pea21] based on the four extreme events that XEventNet predicts. We labeled extreme events for training using the IMD [IMD24] classification of the event. We assigned labels based on the grid point nearest to the recorded event location. A  $10 \times 10$  grid was extracted around this point to maintain spatial consistency. We labeled regions separately for overlapping events such as cyclone and heavy rainfall. Thus they formed the training set for both events. Events are independently labeled for training based on the real data [IMD24] to develop a robust CNN model.

Our simulation domain extends from  $44.42^\circ\text{E}$  to  $127.12^\circ\text{E}$  and  $18.26^\circ\text{S}$  to  $43.38^\circ\text{N}$ . Figure 3 depicts the simulation domain. WRF simulation resolution was set to 9 km for high fidelity simulation. Thus the data size of all variables for one time step is 1.25 GB. The output frequency was set to 1 simulation hour. We ran simulations for 4 days (96 hours) per event for prediction evaluations.

Table 1: WRF Output Variables and Their Descriptions

Variable	Description
P	Perturbation pressure
U	U-component of wind
V	V-component of wind
W	W-component of wind
T	Perturbation temperature
QVAPOR	Water vapor mixing ratio
PB	Base state pressure
PSFC	Surface pressure
SNOWH	Physical snow depth
SNOW	Snow water equivalent
SNOWNC	Accumulated total grid-scale snow and ice
SNOWC	Flag indicating snow coverage
OLR	Outgoing longwave radiation
RAINNC	Accumulated grid-scale precipitation
RAINC	Accumulated cumulus precipitation
RAINSH	Accumulated shallow cumulus precipitation
RH	Relative humidity
SST	Sea surface temperature
U10	U at 10-meter
V10	V at 10-meter

## 4.2. Data Streaming

ADIOS2 SST (Section 2.2) is used for data streaming in XEventNet from (1) simulation to prediction processes and (2) prediction to visualization processes (Figure 1). The streaming happens from memory to memory of the producers and consumers as the simulation runs (*in situ* [Cea20]). XEventNet transfers a fixed volume of data (variables listed in Table 1) in the first case, while the data volume varies for the second transfer. This is because we transfer the most relevant sub-domains only for visualization based on the predictions (Section 3.4). The pseudocode for data streaming is explained next. ADIOS2 SST uses writers at the producer processes and readers at the consumer processes.

XEventNet uses `dlsym` to add a wrapper to the NetCDF function calls. Thus we intercept the data from WRF output function calls at every output time step without changing WRF source code. Listing 1 shows the pseudocode to integrate WRF with ADIOS2 SST functions. The `adios2_begin_step()` function initiates a new step in the ADIOS2 workflow to prepare data for transfer. The `adios2_put()` function stages the generated data for transfer in memory. This is a non-blocking operation. The `adios2_end_step()` function finalizes the current step, making the staged data available for transfer to the reader(s). ADIOS simulation processes (writers) send this data (variables of our interest listed in Table 1) to the prediction processes.

```
if (simulation_is_running)
{
    simulation_output_timestep();
    adios2_begin_step();
    adios2_put(vara_metadata, variable, var_size);
    adios2_end_step();
}
```

Listing 1: Write Data Using ADIOS SST

```
while (data_is_available_for_reading)
{
    adios2_set_selection(vara_metadata, local_start, local_count);
    adios2_get(var_metadata, variable);
    adios2_perform_gets();
    extreme_event_predictions(variable, local_count);
}
```

Listing 2: Read Data Using ADIOS SST

Listing 2 shows the ADIOS hooks on the prediction processes to receive data for further processing. The prediction processes continuously monitors as long as data is produced by the simulation (writers). The function `adios2_set_selection()` specifies the portion of data to be read by each process. The `adios2_get()` retrieves a specific portion of data that has been made available by the producer. The transport layer is responsible for moving this data from where it is staged (e.g., in memory or on a remote system) to the reader requesting it. This is a non-blocking operation. The `adios2_perform_gets()` function ensures that the requested data is transferred to local memory. This step fetches the staged data. The `extreme_event_predictions()` function processes the received data to predict extreme events. The full code is available in [xev25]. We now describe the experimental setup and results.

## 5. Experiments and Results

We first describe the experimental setup followed by results.

### 5.1. Setup

We ran WRF simulation [WRF24] on 32, 64, 128, 256, and 512 processes (1, 2, 4, 8, 16 nodes) on an Intel Xeon Cascade Lake based supercomputer connected via Infiniband 100 Gbps interconnect. Each node has 48 cores, we ran the simulation with 32 MPI ranks per node, thus leaving up to 16 cores for inference on each node. We ran the XEventNet prediction with 1, 2, 4, 8 and 16 processes per node. ADIOS streams data from simulation to prediction processes. Parallel visualization was also done on the same system on 1 – 64 processes. XEventNet was launched as a single job, we mapped the prediction and visualization processes such that the simulation and prediction processes run on different cores of the same node and visualization processes execute on different nodes.

The ratio of the number of simulation to prediction processes plays a significant role in the end-to-end throughput of XEventNet. We demonstrate scalability and efficiency of XEventNet using varying ratios of simulation to prediction processes. The output frequency was kept to 1 hour. The wall-clock time between two output time steps varies from 184 – 12 seconds on 64 and 512 simulation processes respectively. Thus the number of prediction processes has to be selected such that the inference is completed before the next output time step is written by ADIOS.

Figure 3 shows the location of all the extreme events (320) used for training. We generated this dataset using WRF simulation. We plan to make this data available for download. These events were selected from the repository of extreme events during the last 17 years, maintained by IMD [IMD24]. These events comprise severe cyclones (less than 950 mbar pressure), landslides (maximum precipitation of 212.5 mm), snowfall (maximum height of 1.6 m), and heavy rainfall (maximum precipitation 218.3 mm). There are ap-

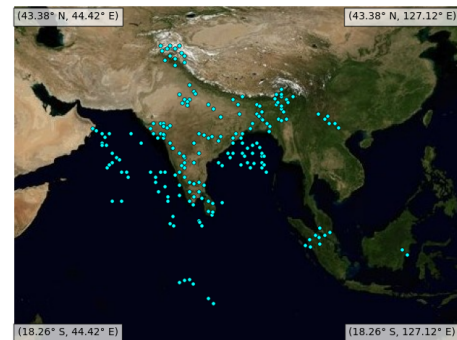


Figure 3: Extreme Weather Events (Tropical Cyclones, Heavy Rainfall, Landslides and Snowfall) Evaluated between 2007 to 2024.

proximately equal number of cyclones, heavy rainfall, landslides and snowfall events in this dataset. We simulated these using WRF and tested the accuracy of XEventNet. We used a ratio of 4:1 for extreme and normal events for training XEventNet. For every extreme event, we used 80% training data and 20% test data.

### 5.2. Results

In this section, we demonstrate performance, scalability, efficacy and utility of XEventNet using various data transfer metrics, CNN scalability, end-to-end times and visualizations.

### 5.2.1. XEventNet Prediction Accuracy

Event	TP	TN	FP	FN	Accuracy	Confidence Score
Cyclone	47.81	46.14	3.85	2.18	0.925	0.861
Heavy Rainfall	47.60	47.18	2.81	2.39	0.948	0.864
Landslide	45.72	43.54	6.45	4.27	0.893	0.801
Snowfall	43.64	41.45	8.54	6.35	0.851	0.798

Table 2: 5-Fold CNN Model Results for all Four Extreme Events.

Table 2 shows the percentages of true positives (TP), true negatives (TN), false positives (FP), false negatives (FN), accuracy and confidence scores (columns 2 – 7) for four different types of extreme weather events (Column 1). These results are averaged from five-fold cross validation runs. These values are calculated at the granularity of a time step. If a time step contains an extreme event at a location, we consider the prediction as positive if XEventNet is able to correctly identify the event at the location. We trained the model on a total of 320 extreme events with an equal distribution of 80 events per extreme event (tropical cyclones, heavy rainfall, landslides, snowfall) across each of the extreme events and another 80 non-extreme events. We observe more than 85% accuracy for all extreme events. The model shows more than 92% accuracy for tropical cyclones and heavy rainfall. We also observe overall low false positives and false negatives for all events.

The results show that XEventNet is able predict multiple extreme events with good accuracy and high confidence. In contrast, a simple threshold based detection method [MM22] exhibits FAR above 60%. The average values of Sensitivity, Specificity, Precision, and F1 Score are 0.92, 0.89, 0.89 and 0.90 respectively. The evaluations across all extreme weather events highlight XEventNet’s efficacy across each event type. Note that XEventNet is also able to predict multiple simultaneously occurring events because we perform the inference in parallel and the same model is trained to identify from among multiple classes of events. Existing models [YCTC22, MM22, ZLHL22, MNSU18] used for extreme event prediction results in POD between 0.35 – 0.97 (for one or two events), which is comparable to XEventNet’s performance (0.87 – 0.96 across four events). Although existing models predict one type of event, they exhibit false alarm ratios between 0.10 – 0.77. In contrast, XEventNet achieves a lower FAR (0.06 – 0.16), thus exhibiting improved reliability with high detection rates.

### 5.2.2. End-to-End XEventNet Performance

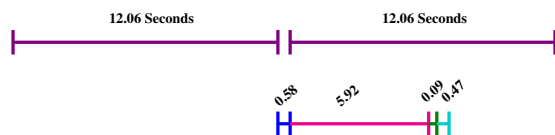


Figure 4: Different stages of our workflow (time in seconds) for 512 simulation processes, 64 prediction and 64 visualization processes.

Figure 4 shows the different stages of our workflow for 512 simulation processes and 16 prediction and visualization processes. These are average times in seconds from five runs of various events. XEventNet workflow comprises of five stages: simulation (purple), streaming (blue), prediction/inference (magenta), transfer (green), and visualization (cyan). Data streaming blocks the simulation and

prediction processes, however the simulation processes are not blocked once the streaming is completed. The simulation processes continue simulating the next time step and prediction processes continue computations simultaneously. It takes 12.06 seconds to simulate 60 simulation minutes on 512 processes and 0.58 seconds to stream the data. It takes 5.92 seconds to predict for a time step on 16 processes, 0.09 seconds to transfer and 0.47 seconds for visualization on 16 processes. Thus the prediction and visualization processes are idle till the next output time step (about 3 seconds in this case). This is a representative figure, we use the appropriate number of simulation and prediction processes to reduce the idle time. We demonstrate scaling results next.

Figure 5 shows the overall scaling of every component of XEventNet. We observe decrease in all times with increase in process count. Figure 5 shows the inference (prediction) time in magenta, data streaming time (from simulation to prediction processes) in blue, simulation time in purple and visualization time in cyan. These times (average) are shown for one time step. The simulation process count is shown in the X-axis (bottom labels), this varies from 32 – 512 processes with 32 ranks per node, i.e. 1 – 16 nodes. The total prediction or inference process count and the visualization process count is shown at the top (varies from 4 – 64 processes). We have used a ratio of 8:1 between simulation and prediction processes on each node. Thus, we schedule 32 simulation processes and 4 prediction processes per node for all the configurations. The visualization processes execute on separate nodes in the same supercomputer. Y-axis shows the time in seconds per time step. All times shown in this figure are an average of five runs from different events. The variation across runs for inference time is 0.45 seconds whereas it is less than 0.1 seconds for other times.

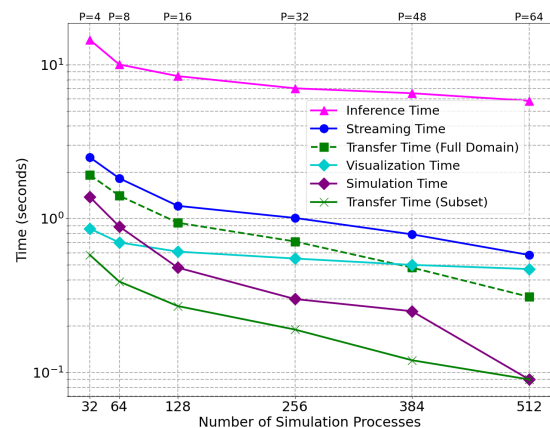


Figure 5: XEventNet Strong Scaling: Simulation processes vary from 32 – 512 processes, prediction and visualization processes vary from 4 – 64 processes.

Figure 5 demonstrates strong scaling of XEventNet – we note that almost all times scale well. The simulation time decreases from 1.38 s per time step for 32 processes to 0.09 s per time step for 512 processes. Decrease in simulation time implies less time between two output time steps, during which the inference and streaming should be completed for the previous time step. The output frequency was 1 simulation hour. This corresponds to 134 simulation timesteps. Thus the time between two output time steps (prediction frequency) is 184.92 s for 32 processes whereas it is

12.06 s for 512 processes. The prediction time decreases from 14.51 s for 4 processes to 5.83 s for 64 processes due to parallel CNN in XEventNet.

It is necessary to appropriately scale up the prediction processes as we scale up the simulation processes. For every configuration, 8 simulation processes stream data to 1 prediction process using ADIOS SST APIs. The data size per stream decreases due to strong scaling from 0.04 GB per stream at 32 processes to 0.002 GB per stream at 512 processes. Thus, the streaming time also decreases with increase in simulation and prediction processes from 2.5 seconds at 32 processes to 0.58 seconds at 512 processes. In future, we plan to leverage GPUs for some parts of the workflow. We focused on end-to-end scalability despite the differences in compute rates of parallel simulation, inference and visualization avoiding I/O completely in this work. The time also decreases because the multiple data streams occur on different nodes concurrently.

The simulations were run for a period of 1 day (24 simulation hours). The data streaming frequency (simulation output frequency) was 1 simulation hour. Thus the total number of output time steps was 24. Every hour, 0.8 GB data was streamed from simulation to prediction processes for all five configurations shown in the figure. This amounts to a total data size of 19.2 GB corresponding to the 20 variables used for predictions in XEventNet. After the predictions, XEventNet transfers data based on positive predictions. Thus the data size for data streamed from prediction to visualization processes is less than 0.8 GB per output time step. In our case, it was 0.14 GB on average (ranging from 0.03 – 0.25 GB). Figure 5 shows transfer time in green solid line for our reduced data transfer approach for visualization and transfer time in green dashed line if we transferred the full domain. The transfer time from prediction to visualization processes using XEventNet’s reduced data approach (Section 3.4) is shown in green dotted line across all runs. The total data size transferred using full domain is 19.2 GB, whereas XEventNet transfers 3.3 - 3.7 GB for 24 time steps.

We note both transfer times (full domain and subset of domains) decrease with increase in the number of prediction and visualization processes from 4 to 64. This is because of increase in the number of parallel data transfers between the prediction and visualization processes. The transfer time with the reduced data approach in XEventNet is almost an order of magnitude less than the transfer time for full domain. Although we transfer a subset of domains, we overlay (patch) the new data on the data from the previous time step at the visualization processes. Thus we visualize the full domain despite transferring a subset of it. Traditional post hoc approach takes a total of 508.92s (343s for simulation that includes write times and 164.92s for inference) to predict offline from data written by simulation (512 processes) for 24 time steps. In contrast, XEventNet takes 294s by avoiding I/O and using concurrent parallel inference from sub-domains.

### 5.2.3. XEventNet CNN Scalability

We show the scalability of in situ parallel CNN model in Table 3, i.e. XEventNet prediction model runs while WRF simulation is running on 128, 256, 512 processes (4, 8, and 16 nodes respectively). Column 1 shows the number of prediction processes (4 – 64). These are distributed across 4 nodes, 8 nodes and 16 nodes in case of 128,

256, 512 processes. For each process count, we show the streaming times from 128, 256, 512 simulation processes for each prediction process count in columns 2, 3, 4 respectively. We observe decrease in streaming times as we increase the prediction process count (top to bottom) and as we increase the simulation process count (column 2 – 4). The best streaming time of 0.58 s is obtained at 512 simulation processes and 64 prediction processes, which enables simultaneous data streaming of 0.8 GB data from 512 to 64 prediction processes. This corresponds to an effective data bandwidth of 11.03 Gbps. Column 5 shows the inference times for the most optimal batch size. We observe decrease in inference times from 14.68 s at 4 processes to 5.89 s at 64 processes. We will show detailed results for effect of batch size next.

Process Count	Streaming Time (s)			Inference Time (s)	Transfer Time (s)
	128	256	512		
4	2.38	1.99	1.18	14.67	0.58
8	1.85	1.57	1.04	10.19	0.39
16	1.55	1.21	0.83	8.44	0.27
32	1.42	1.01	0.69	7.17	0.19
64	1.24	0.87	0.58	5.92	0.09

Table 3: Strong scaling on 4 – 64 prediction processes.

Figure 5 shows the scaling of inference time using an optimal batch size based on the number of simulation (writers) and prediction processes (readers). We observed scalability saturation with a constant batch size, thus we empirically evaluated different batch sizes to derive the optimal value. This is because the data size per process for 4 prediction process is 16× larger than the data size per process for 64 prediction processes. Thus the optimal batchsize for each configuration varies. Figure 6 shows the scaling of inference

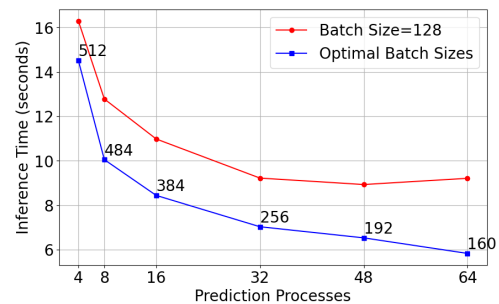


Figure 6: Scaling of inference time with a constant batch size of 128 vs. optimal batch size for 4 – 64 prediction processes.

time on 4 – 64 prediction processes for two variations of batch size. The red curve shows the prediction times using a constant batch size of 128. The blue curve shows the inference times using an optimal batch size for the given prediction process count. The optimal batch size is annotated on the curve. More kernel invocations occur for higher batch sizes. The batch size decreases from 4 – 64 because the data size per batch decreases due to increase in number of prediction processes, thus it suffices to have lower batch sizes for higher process counts.

Number of Processes	1	2	4	8	16	32
Training Time (Hours)	8.05	5.55	3.39	2.11	1.13	0.64

Table 4: Training time scaling with varying numbers of processes.

Table 4 shows strong scaling results of training times. We demonstrate the times on 1 – 32 processes (one node) for training the model on 5000 training points. These data points are the  $10 \times 10$  grids of the domains (Section 3.2). We kept the number of epochs constant at 10 for all configurations. The training time decreases significantly from 8.05 hours on 1 process to 38.6 minutes on 32 processes for the same number of training points as we increase the number of processes. This demonstrates effectiveness of distributed training in XEventNet. Efficiency decreases from 0.7 at lower process counts to 0.4 at larger process counts due to possible communication overhead. We plan to explore this further in future.

#### 5.2.4. Data Transfer Volume and Visualized Data

Figure 7 shows information about the data volume transferred from 64 prediction processes to the visualization processes. X-axis shows the number of subdomains transferred across various time steps for a 24-hour simulation run of three extreme events. The total data size per time step was 800 MB, thus the total data size for each subdomain is 12.5 MB per prediction process. The number of subdomains eventually transferred by XEventNet depend on the output of the CNN classification that is performed in parallel on the 64 processes. This depends on whether the subdomain is predicted to contain an extreme event or not. Figure 7 shows variation in the number of transferred subdomains (0 – 19). A value of 0 implies that no subdomains were selected during that time step, due to absence of extreme events across all subdomains. Thus the data size transferred for visualization varied from 0 – 238 MB. The maximum amount of data selected is less than 30% of the total data. This reduces the transfer time by more than 74% per time step. This is essential to visualize data at a user-desired high frequency. A 24-hour simulation with a prediction/output frequency of 1 hour result in 19.2 GB data for all the 20 variables (Table 1). Across all our test prediction runs, the average data transferred for cyclone, heavy rainfall, landslide and snowfall was 2.89 GB, 4.05 GB, 3.05 GB and 3.63 GB respectively.

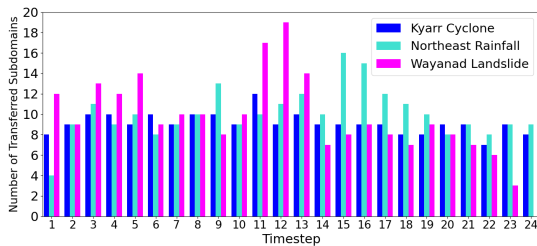


Figure 7: Number of subdomains selected for each time step for one day simulation of three events: Cyclone Kyarr November 2019, Northeast Rainfall June 2024, and Wayanad Landslide July 2024.

A popular data reduction methodology is compression. However this entails compressing the data and decompressing. We compared the performance of using compression in place of transferring subset of data. We found that popular compression algorithms such as gzip, lz4 and zstandard [KAP15, KKKU17, CK18] take more than our allowable budget for data transfer. gzip takes 2.21 s to compress 800 MB to 730 MB and about 1 s to decompress it. lz4 takes less than a second to compress and decompress, however, the compression ratio was close to 1. zstandard takes 1.21 s to compress and 0.63 s to decompress with a compression ratio of close to 1. Thus our approach not only demonstrates higher compression ratio but also

does not add any compression/decompression overhead. This is because the sub-domain data selection for transfer is automatically implied by the inference processes.

Figures 8, 9, and 10 show the visualization of three extreme events – cyclone Fani in the location (18.32, 85.01) during May 2019, landslide event in the location (11.70, 76.08) during July 2024, and heavy snowfall in the location (34.73, 76.18) during March 2016 respectively. The figures show colour plot of variables  $P$  (Perturbation pressure) for cyclone,  $RAIN_C$  (Accumulated cumulus precipitation) for landslide and  $SNOWNC$  (Accumulated total grid scale snow and ice) for snowfall. The magenta rectangles on the figures denote the subdomains that were transferred for visualization from prediction processes. The rest of the subdomains are visualized after patching (overlying) with the previously transferred time step as explained in Section 3.4. These subdomains had an average of 0.8 confidence score for at least one extreme event in the  $10 \times 10$  grids of the subdomain. Observe that the CNN model predicts regions with extreme events with high confidence and these regions overlap with the actual extreme event locations in all cases. This demonstrates the importance and utility of XEventNet. Figure 11 shows the exact same time step as shown in Figure 10, however when the timestep was transferred completely. We observe hardly any visual differences between the two frames. The cosine similarity value of the visualized variable SNOWNC is 0.95, thereby quantitatively indicating high similarity between the two frames despite the fact that one of them is generated using our overlaid approach. This demonstrates the efficacy of XEventNet.

## 6. Conclusions and Future Work

In this work, we proposed an online integrated framework, XEventNet, for simulation, prediction and visualization of extreme weather events. We developed a distributed CNN-based prediction model for extreme event weather prediction of four extreme events – tropical cyclones, heavy rainfall, landslides and snowfall. XEventNet performs parallel data streaming from an ongoing weather simulation for online extreme event prediction followed by in situ visualization of the same. Parallel training reduced the time from 8.05 hours on 1 process to 0.64 hours on 32 processes for training 5000 data points. Parallel inference time decreased from 26.18 s on 1 process to 5.92 s on 64 processes.

We used ADIOS2 library for efficient data streaming via system memory (instead of low-bandwidth secondary storage). Note that inference and visualization times are reported from the online XEventNet runs. The prediction processes selectively transfer only a subset of the whole domain based on the positive predictions. This reduces the data transfer times by more than 70% which is crucial in our online XEventNet framework.

In future, we plan to extend to more extreme events and perform multi-event visualization using the multiple rendering windows. The CNN model of XEventNet may be used for these four events (of the same intensity) across other geographical regions. This requires executing WRF to generate test data for those regions. This is our future work. We also plan to test the scalability of other distributed deep learning frameworks such as Horovod and PyTorch. Further, we plan to perform the predictions and visualizations on GPUs and perform a cost vs. performance vs. energy comparisons.

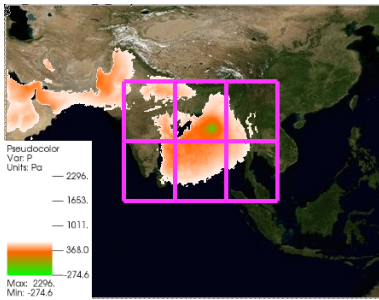


Figure 8: Visualization of variable P for Fani Cyclone event on 2<sup>nd</sup> May 2019 at 6 PM IST. Time step number is 1206.

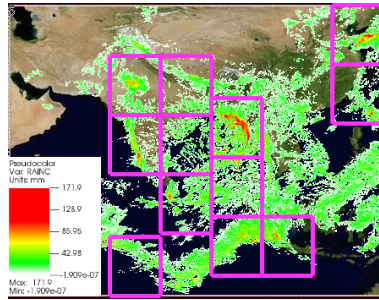


Figure 9: Visualization of variable RAINC for Wayanad Landslide event on 30<sup>th</sup> July 2024 at 4 AM IST. Time step number is 536.

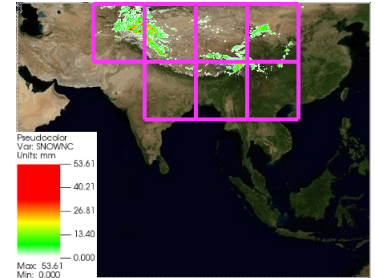


Figure 10: Visualization of variable SNOWNC for Kargil Snowfall Avalanche event on 19<sup>th</sup> March 2016 at 8 AM IST. Time step number is 1088.

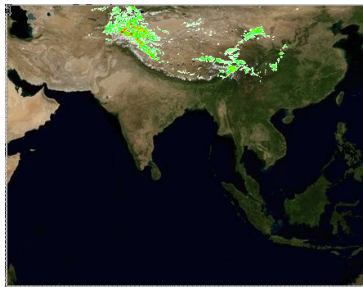


Figure 11: Visualization of SNOWNC for Snowfall Avalanche event on 19<sup>th</sup> March 2016 at 8 AM IST using the full data transfer. Time step number visualized is 1088.

### Acknowledgements

We thank the Computer Center and the Department of Computer Science and Engineering, IIT Kanpur for enabling 24 × 7 access to computer resources. We acknowledge National Supercomputing Mission (NSM) for providing computing resources of PARAM Sanganak at IIT Kanpur.

### References

[ABG\*15] AYACHIT U., BAUER A., GEVECI B., O’LEARY P., MORELAND K., FABIAN N., MAULDIN J.: ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (2015), p. 25–29. 3

[Aea25] AHRENS J., ET AL.: The ECP ALPINE project: In situ and post hoc visualization infrastructure and analysis capabilities for exascale. *The International Journal of High Performance Computing Applications* 39, 1 (2025), 32–51. 3

[AF18] AVOLIO E., FEDERICO S.: WRF simulations for a heavy rainfall event in southern Italy: Verification and sensitivity tests. *Atmospheric Research* 209 (2018), 14–35. 5

[AM20] AGRAWAL A., MITTAL N.: Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. *Vis. Comput.* 36, 2 (Feb. 2020), 405–412. 4

[BXZ\*22] BI K., XIE L., ZHANG H., CHEN X., GU X., TIAN Q.: Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast, 2022. [arXiv:2211.02556](https://arxiv.org/abs/2211.02556). 2, 3

[BXZ\*23] BI K., XIE L., ZHANG H., CHEN X., GU X., TIAN Q.: Accurate medium-range global weather forecasting with 3D neural networks. *Nature* 619, 7970 (2023), 533–538. 1, 2

[Cea20] CHILDS H., ET AL.: A terminology for in situ visualization and analysis systems. *The International Journal of High Performance Computing Applications* 34, 6 (2020), 676–691. 2, 6

[CHG\*23] CHEN K., HAN T., GONG J., BAI L., LING F., LUO J.-J., CHEN X., MA L., ZHANG T., SU R., CI Y., LI B., YANG X., OUYANG W.: FengWu: Pushing the Skillful Global Medium-range Weather Forecast beyond 10 Days Lead, 2023. [arXiv:2304.02948](https://arxiv.org/abs/2304.02948). 1, 2

[CK18] COLLET Y., KUCHERAWY M. S.: Zstandard Compression and the application/zstd Media Type. *RFC 8878* (2018), 1–45. 9

[CL22] CHEN J., LIU Z.: An Efficient Parallel CNN Inference Framework for Multi-zone Processor. In *2022 IEEE 24th Int Conf on High Performance Computing and Communications* (2022). 2

[DBE\*14] DAYAL J., BRATCHER D., EISENHAEUER G., SCHWAN K., WOLF M., ZHANG X., ABBASI H., KLASKY S., PODHORSZKI N.: Flexpath: Type-Based Publish/Subscribe System for Large-Scale Science Analytics. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (2014), pp. 246–255. 3

[DP17] DREHER M., PETERKA T.: Decaf: Decoupled Dataflows for In Situ High-Performance Workflows. URL: <https://www.osti.gov/biblio/1372113>, doi:10.2172/1372113. 3

[DPK10] DOCAN C., PARASHAR M., KLASKY S.: DataSpaces: an interaction and coordination framework for coupled simulation workflows. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing* (2010), p. 25–36. 3

[Eea24] EISENHAEUER G., ET AL.: Streaming Data in HPC Workflows Using ADIOS, 2024. [arXiv:2410.00178](https://arxiv.org/abs/2410.00178). 3

[FDJ\*23] FREDJ E., DELORME Y., JUBRAN S., WASSERMAN M., DING Z., LAUFER M.: Accelerating WRF I/O Performance with ADIOS2 and Network-based Streaming, 2023. 3

[FGVS\*15] FERNÁNDEZ-GONZÁLEZ S., VALERO F., SÁNCHEZ J. L., GASCÓN E., LÓPEZ L., GARCÍA-ORTEGA E., MERINO A.: Numerical simulations of snowfall events: Sensitivity analysis of physical parameterizations. *Journal of Geophysical Research: Atmospheres* 120, 19 (2015), 10,130–10,148. 5

[FISA22] FAN M., IMRAN O., SINGH A., AJILA S. A.: Using CNN-LSTM Model for Weather Forecasting. In *2022 IEEE International Conference on Big Data (Big Data)* (2022), pp. 4120–4125. 2

[FYS\*19] FU J., YANG Y., SINGHRAO K., RUAN D., CHU I., LOW D. A., LEWIS J. H.: Deep learning approaches using 2D and 3D convolutional neural networks for generating male pelvic synthetic computed tomography from magnetic resonance imaging. *Medical Physics* 46, 9 (2019), 3788–3798. 4

[Gea20] GODOY W. F., ET AL.: ADIOS 2: The Adaptable Input Output System. A framework for high-performance data management. *SoftwareX* 12 (2020), 100561. 2, 3

[GJGP22] GIRI R. N., JANGHEL R. R., GOVIL H., PANDEY S. K.: Spatial Feature Extraction using Pretrained Convolutional Neural network for Hyperspectral Image Classification. In *2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)* (2022), pp. 386–389. 4

- [Gra79] GRAY W. S.: Hurricanes: Their formation, structure and likely role in the tropical circulation. URL: <https://api.semanticscholar.org/CorpusID:132807346>. 5
- [IMD24] IMD: Indian Meteorological Department, 2024. URL: <https://mausam.imd.gov.in/>. 1, 3, 5, 6
- [Ind15] INDIA I. P.: Disastrous Weather Events, 2015. 1
- [KAP15] KALAJDZIC K., ALI S. H., PATEL A.: Rapid lossless compression of short text messages. *Computer Standards & Interfaces* 37 (2015), 53–59. 9
- [KCTA24] KHAIRA U., CERRAI D., THOMPSON G., ASTITHA M.: Integrating physics-based WRF atmospheric variables and machine learning algorithms to predict snowfall accumulation in Northeast United States. *Journal of Hydrology* 644 (2024), 132113. 3
- [KKKU17] KHAN A., KHAN A., KHAN M., UZAIR M.: Lossless image compression: application of Bi-level Burrows Wheeler Compression Algorithm (BBWCA) to 2-D data. *Multimedia Tools and Applications* 76, 10 (2017), 12391–12416. 9
- [KSH\*23] KURTH T., SUBRAMANIAN S., HARRINGTON P., PATHAK J., MARDANI M., HALL D., MIELE A., KASHINATH K., ANANDKUMAR A.: FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference* (2023). 1, 2, 3
- [Lea16] LIU Y., ET AL.: Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets, 2016. [arXiv:1605.01156](https://arxiv.org/abs/1605.01156). 2
- [Lea19] LEE S., ET AL.: Improving Scalability of Parallel CNN Training by Adjusting Mini-Batch Size at Run-Time. In *2019 IEEE International Conference on Big Data (Big Data)* (2019), pp. 830–839. 5
- [LF22] LAUFER M., FREDJ E.: High Performance Parallel I/O and In-Situ Analysis in the WRF Model with ADIOS2, 2022. 3
- [LJ14] LEHMANN H., JUNG B.: In-situ multi-resolution and temporal data compression for visual exploration of large-scale scientific simulations. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)* (2014), pp. 51–58. 3
- [MM22] MUKHERJEE A., MALAKAR P.: A Deep Learning-Based In Situ Analysis Framework for Tropical Cyclogenesis Prediction. In *2022 IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC)* (2022), pp. 166–175. 2, 5, 7
- [MNSU18] MATSUOKA D., NAKANO M., SUGIYAMA D., UCHIDA S.: Deep learning approach for detecting tropical cyclones and their precursors in the simulation by a cloud-resolving global nonhydrostatic atmospheric model. *Progress in Earth and Planetary Science* 5, 1 (dec 2018), 80. 2, 7
- [NBK\*23] NGUYEN T., BRANDSTETTER J., KAPOOR A., GUPTA J. K., GROVER A.: ClimaX: A foundation model for weather and climate, 2023. [arXiv:2301.10343](https://arxiv.org/abs/2301.10343). 2
- [Pea21] PERMANA D. S., ET AL.: Evaluation of different WRF microphysics schemes: a case study of landslide induced by heavy rainfall in Kulon Progo. *IOP Conference Series: Earth and Environmental Science* 874, 1 (oct 2021), 012009. 5
- [PVH\*03] POST F., VROLIJK B., HAUSER H., LARAMEE R., DOLEISCH H.: The State of the Art in Flow Visualisation: Feature Extraction and Tracking. *Comput. Graph. Forum* 22 (12 2003), 775–792. 3
- [QHDT20] QUANG-HUNG N., DOAN H., THOAI N.: Performance Evaluation of Distributed Training in Tensorflow 2. In *2020 International Conference on Advanced Computing and Applications (ACOMP)* (2020), pp. 155–159. 2
- [QZZ\*17] QIU M., ZHAO P., ZHANG K., HUANG J., SHI X., WANG X., CHU W.: A Short-Term Rainfall Prediction Model Using Multi-task Convolutional Neural Networks. In *2017 IEEE International Conference on Data Mining (ICDM)* (2017), pp. 395–404. 1, 2
- [RGGGM19] RAMIREZ-GARGALLO G., GARCIA-GASULLA M., MANTOVANI F.: TensorFlow on State-of-the-Art HPC Clusters: A Machine Learning use Case. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)* (2019), pp. 526–533. 4
- [SCW\*15] SHI X., CHEN Z., WANG H., YEUNG D.-Y., WONG W.-K., WOO W.-C.: Convolutional LSTM Network: a machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Cambridge, MA, USA, 2015), NIPS’15, MIT Press, p. 802–810. 2
- [SEH\*20] SØNDERBY C. K., ESPEHOLT L., HEEK J., DEGHANI M., OLIVER A., SALIMANS T., AGRAWAL S., HICKEY J., KALCHBRENNER N.: MetNet: A Neural Weather Model for Precipitation Forecasting. *CoRR abs/2003.12140* (2020). 1, 2, 3
- [Ser24] SERVICE N. W.: Weather-Related Fatality and Injury Statistics. <https://www.weather.gov/hazstat/>, 2024. 1
- [SKD\*08] SKAMAROCK W., KLEMP J., DUDHIA J., GILL D., BARKER D., WANG W., POWERS J.: A Description of the Advanced Research WRF Version 3. 3–27. 2, 5
- [Sto16] STOTT P.: How climate change affects extreme weather events. *Science* 352, 6293 (2016), 1517–1518. 1
- [TSP\*24] TRIVEDI D., SHARMA O., PATNAIK S., HAZRA V., PUHAN N. B.: Improving rainfall forecast at the district scale over the eastern Indian region using deep neural network. *Theoretical and Applied Climatology* 155, 1 (jan 2024), 761–777. 1, 2
- [UCL24] UCLouvain BRUSSELS B.: The International Disaster Database. <https://www.emdat.be/>, 2024. 1
- [WFM11a] WHITLOCK B., FAVRE J. M., MEREDITH J. S.: Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization* (Goslar, DEU, 2011), EGPGV ’11, Eurographics Association, p. 101–109. 3
- [WFM11b] WHITLOCK B., FAVRE J. M., MEREDITH J. S.: Parallel in Situ Coupling of Simulation with a Fully Featured Visualization System. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization* (Goslar, DEU, 2011), EGPGV ’11, Eurographics Association, p. 101–109. 5
- [WRF24] WRF DEVELOPERS: Weather Forecasting Model (WRF) code. <https://github.com/wrf-model/WRF>, 2024. 6
- [WT22] WANG F., TIAN D.: On deep learning-based bias correction and downscaling of multiple climate models simulations. *Climate Dynamics* 59, 11 (2022), 3451–3468. 2
- [xev25] XEVENTNET: XEventNet code. <https://github.com/MuzafarWani/xeventnet>, 2025. 6
- [YCTC22] YAO S., CHEN H., THOMPSON E. J., CIFELLI R.: An Improved Deep Learning Model for High-Impact Weather Nowcasting. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2022), 7400–7413. 7
- [ZCF\*24] ZHONG X., CHEN L., FAN X., QIAN W., LIU J., LI H.: FuXi-2.0: Advancing machine learning weather forecasting model for practical applications, 2024. [arXiv:2409.07188](https://arxiv.org/abs/2409.07188). 2
- [ZLHL22] ZHANG R., LIU Q., HANG R., LIU G.: Predicting Tropical Cyclogenesis Using a Deep Learning Method From Gridded Satellite and ERA5 Reanalysis Data in the Western North Pacific Basin. *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022). 7