

PAPER • OPEN ACCESS

## *In situ* compression artifact removal in scientific data using deep transfer learning and experience replay

To cite this article: Sandeep Madireddy *et al* 2021 *Mach. Learn.: Sci. Technol.* **2** 025010

View the [article online](#) for updates and enhancements.

You may also like

- [A robotic model of hippocampal reverse replay for reinforcement learning](#)  
Matthew T Whelan, Alejandro Jimenez-Rodriguez, Tony J Prescott et al.
- [An outlier detection-based method for artifact removal of few-channel EEGs](#)  
He Chen, Hao Zhang, Chuancai Liu et al.
- [Signal processing methods for reducing artifacts in microelectrode brain recordings caused by functional electrical stimulation](#)  
D Young, F Willett, W D Memberg et al.



## PAPER

# *In situ* compression artifact removal in scientific data using deep transfer learning and experience replay


## OPEN ACCESS

RECEIVED  
24 August 2020REVISED  
23 September 2020ACCEPTED FOR PUBLICATION  
20 October 2020PUBLISHED  
29 December 2020

Original Content from  
this work may be used  
under the terms of the  
[Creative Commons  
Attribution 4.0 licence](#).

Any further distribution  
of this work must  
maintain attribution to  
the author(s) and the title  
of the work, journal  
citation and DOI.



Sandeep Madireddy<sup>1</sup> , Ji Hwan Park<sup>2</sup>, Sunwoo Lee<sup>3</sup>, Prasanna Balaprakash<sup>1</sup>, Shinjae Yoo<sup>2</sup>, Wei-keng Liao<sup>3</sup>, Cory D Hauck<sup>4</sup>, M Paul Laiu<sup>4</sup> and Richard Archibald<sup>4</sup>

<sup>1</sup> Argonne National Laboratory, Lemont, IL, United States of America

<sup>2</sup> Brookhaven National Laboratory, Upton, NY, United States of America

<sup>3</sup> Northwestern University, Evanston, IL, United States of America

<sup>4</sup> Oak Ridge National Laboratory, Oak Ridge, TN, United States of America

E-mail: [smadireddy@anl.gov](mailto:smadireddy@anl.gov)

**Keywords:** compression artifact removal, incremental transfer learning, convolutional neural networks, compressed sensing, hyperbolic PDE systems

## Abstract

The massive amount of data produced during simulation on high-performance computers has grown exponentially over the past decade, exacerbating the need for streaming compression and decompression methods for efficient storage and transfer of this data—key to realizing the full potential of large-scale computational science.

Lossy compression approaches such as JPEG when applied to scientific simulation data realized as a stream of images can achieve good compression rates but at the cost of introducing compression artifacts and loss of information. This paper develops a unified framework for *in situ* compression artifact removal in which the fully convolutional neural network architectures are combined with scalable training, transfer learning, and experience replay to achieve superior accuracy and efficiency while significantly decreasing the storage footprint as compared with the traditional optimization-based approaches.

We demonstrate the proposed approach and compare it with compressed sensing postprocessing and other baseline deep learning models using climate simulations and nuclear reactor simulations, both of which are driven by hyperbolic partial differential equations. Our approach when applied to remove the compression artifacts on the JPEG-compressed nuclear reactor simulation data (using a transfer-trained model that was pretrained on the climate simulation data and updated incrementally as the nuclear reactor simulation progressed), achieved a significant improvement—mean peak signal-to-noise ratio of 42.438 as compared with 27.725 obtained with the compressed sensing approach.

## 1. Introduction

Digital image acquisition and processing have a rich history with active and robust research that has accelerated up to the present day [1]. A major challenge for digital imaging has been compression, a necessary tool for image analysis that has direct impact on the cost of data storage and movement. Traditionally, compression algorithms have been developed to perform optimally for consumer imaging and video needs. A related problem has arisen in the high-performance computing (HPC) science community, where the growth of computational power has enabled an explosion of data from increased spatial and temporal resolution of scientific simulations that has outstripped the ability to store the valuable information produced [2], creating a strong need for compression in this field [3].

The field of study for compression algorithms has a long history and is currently very active [4]. In this work, we consider the scientific data reduction obtained through the application of image compression methods to a stream of images being produced by high-performance simulations. At the rates of compression necessary for large data streams, lossy compression algorithms that maximize compression rates

are preferred, but it comes with the cost of introducing compression artifacts and loss of information. The lossy compression algorithms used for scientific data are either prediction-based or transform-based [5]. While ZFP is the current state of the art in transform-based compression for integer and floating-point scientific data, we adopted JPEG (discrete cosine transform-based) [6] in this work because it is arguably the most popular, and commonly used form of lossy compression [1, 7]. Moreover, since we seek to develop *in situ* machine learning-based approaches to improve the accuracy of lossy compressed data streams, considering a JPEG that is well studied in machine learning literature gives good baselines to build upon. The ability to accurately and efficiently restore large quantities of the lossy-compressed image data generated while storing and transferring scientific simulation data holds the key to realizing the full potential of large-scale computational science [8].

Image compression artifact removal (CAR) (is a example of image restoration which also includes denoising and super-resolution) on single images has also been studied in great detail, with early approaches using handcrafted filters [9], iterative optimization approaches [10], sparse dictionary learning [11]. Among the traditional optimization-based approaches, compressed sensing has shown great potential for image reconstruction and denoising [12–15], regularization, and uncertainty quantification in scientific simulation [16–20]. Compressed sensing algorithmically exploits the sparsity patterns in the data through regularization and thus finds a compressed representation. Several deep learning (DL) approaches that pose this as a supervised learning problem in which a functional mapping is learned between corrupted and restored images by using several training examples have also been proposed [21, 22] and showed notable improvements over the traditional approaches. AR-CNN is one of the early approaches that used a shallow neural network architecture for JPEG artifact removal. Models with deeper architectures [23] and those based on residual networks [24] have also been adopted. Other approaches such as image priors [25] and generative adversarial networks [26] have also been explored in the context of CAR. A different stream of work has addressed the artifact removal in sequential data (e.g. videos), where a temporal correlation exists between frames. A simple assumption made by several works is to ignore the correlation and use the single-image CAR approaches [27]. Other works that exploit the temporal correlation are either seeking to explicitly estimate the motion between the frames [28] or use nonlocal operators and recurrent architectures without motion estimation [29, 30].

In this paper, we focus specifically on artifact removal for JPEG-compressed images from large-scale scientific simulations corresponding to two different domains—climate simulations and nuclear reactor simulation. In contrast to the previous works in CAR, we are interested in the *in situ* training of the CAR model as the simulation progresses, as well as in transfer learning across domains. The goal of the *in situ* training is to mitigate the model training time as well as the need to store full data on disk by storing compressed data and a trained CAR model instead. The transfer learning across domains is motivated by the fact that both the domains are governed by hyperbolic partial differential equations (PDEs), which include a significant fraction of the simulations performed on HPC systems [2]).

To achieve this goal, we propose an incremental transfer learning (based on discrepancy-based deep domain adaptation (DA) [31] along with an experience replay mechanism [32–35]) that efficiently updates the model by using knowledge from previous checkpoints as well as pretrained models from other domains. In this framework, we adopt artifact removal approaches based on DL that employ fully convolutional neural networks (CNNs) to learn the mapping between the images with JPEG compression artifacts and their corresponding uncompressed images. We compare our approach with compressed sensing and other offline DL-based CAR approaches, which are used as baselines. In addition, we compare our approach with DL-based approaches that take into account the full spatiotemporal correlation of the image sequence in a simulation while removing artifacts, which is the best-case scenario in the offline setting.

Although DL-based CAR has been extensively studied in the literature, its application to scientific simulation data is very limited, let alone to transfer learning across scientific domains or *in situ* training. With this work we fill these gaps by making the following contributions:

- We adopt a DL-based CAR approach for scientific simulation data and show that it is an accurate and efficient alternative to traditional optimization-based compressed sensing approaches.
- We propose an incremental batch transfer learning approach that updates the DL-based CAR model *in situ*, simultaneously as the simulation is run, and does so efficiently by transferring knowledge from previously trained models (using data from within and across domains) as well as employing data-parallel training strategies.

## 2. Data—scientific simulations

The scientific simulation community has many reduced models that represent simplified versions of more complicated systems. These models are well studied and form a basis of understanding for application scientists analyzing simulations. In this section, we introduce reduced models for climate and nuclear reactor simulations, two different applications that are actively studied within the scientific computing community.

### 2.1. Climate

The shallow water equations on the sphere are a reduced-order model used to simulate basic dynamics of the Earth's atmosphere. The climate community has developed various test case scenarios to understand the numerical properties of discrete methods for the shallow water equations. Most notable are the original test cases proposed in [36], the moving and stationary vortices on the sphere test [37], the cross-polar flow test [38], and the barotropic instability test [39]. In this paper we consider the barotropic instability test case that is simulated by using a fourth-order Runge–Kutta method [40] for time-stepping and a discontinuous Galerkin (DG) spatial discretization with a Legendre basis on a cubed-sphere mesh [41] for the spatial discretization.

The cubed sphere, first developed in [42], has proven to be a particularly useful gridding technique for solving PDEs on the sphere. In this study we use the shallow water equations on the rotating sphere in flux form, given in [41], on the barotropic instability test defined in [39]. This is a standard test for assessing the quality of atmospheric numerical methods and involves perturbing the atmospheric flow by a localized bump to the balanced height field. In this paper we use three variants of the barotropic instability test, by setting the localized bump to the balanced height field as  $\hat{h} = 120$  m, 180 m, and 240 m.

### 2.2. Nuclear reactor simulation

Kinetic transport equations are used to simulate the movement of particles, such as neutrons in a reactor, that interact with a background material medium. In a three-dimensional geometry, these equations describe the evolution of a distribution function that is defined over a six-dimensional phase space. However, under the assumption that there is zero variation in the vertical direction and the particles have unit speed, the phase space can be reduced to four dimensions. Moreover, if the particles scatter isotropically, then a kinetic transport equation reduces to the form given in [43] that has structural similarities with the shallow water equations. Indeed both are hyperbolic balance laws, in other words, hyperbolic conservation laws with additional source terms. However, substantial differences exist. First, the kinetic equation typically requires more unknowns for accurate simulations than do the shallow water equations. Second, whereas the kinetic transport equation is linear, the shallow water equations are non-linear and thus exhibit a richer evolutionary structure that includes shock waves and rarefactions.

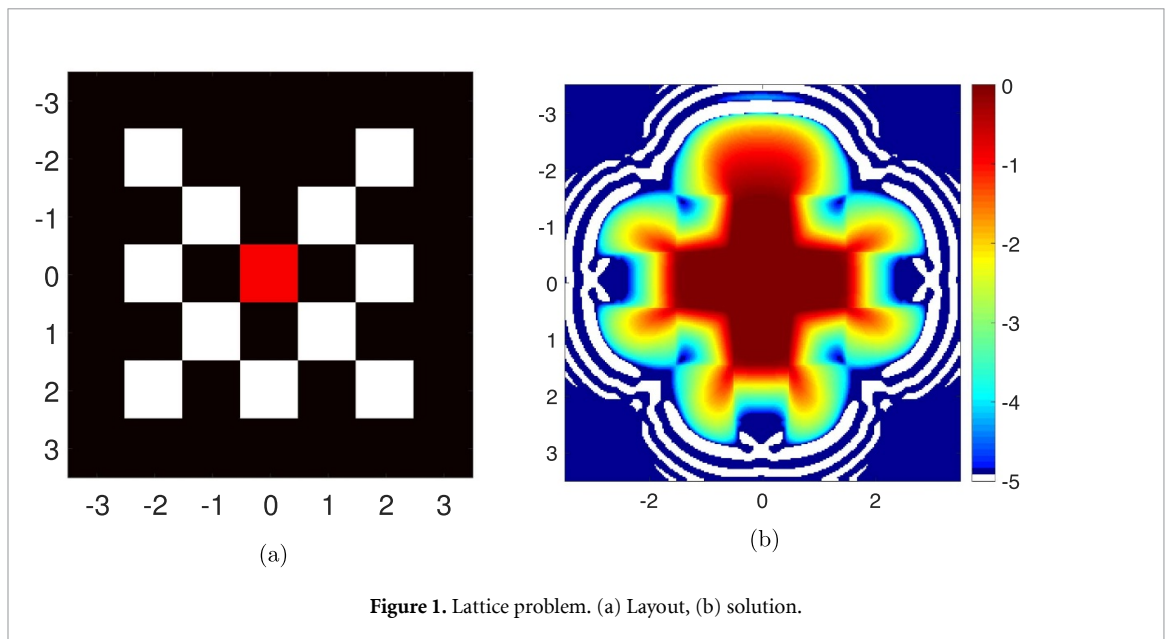
#### 2.2.1. Numerics.

In this paper, we use simulation results based on a spherical harmonic approximation, referred to as the *filtered spherical harmonic equations*. For more details, see [44, 45]. Once derived, it is discretized by using the kinetic scheme from [46, 47]. This is a second-order finite volume method that uses upwind fluxes at the kinetic level but no limiters on the slope reconstructions. The evaluation of the fluxes requires integration, which is performed with a standard tensor product quadrature; see, for example, [48, 49]. The time integrator is an SSP-RK2 method (also known as Heun's method). See [47] for details.

#### 2.2.2. Lattice problem.

The lattice problem, proposed in [50], is based loosely on the geometry of an assembly in a nuclear reactor core. The problem is specified on a two-dimensional spatial domain  $[-3.5, 3.5] \times [-3.5, 3.5]$  that consists of a purely scattering medium with strongly absorbing regions embedded as a checkerboard and an isotropic source at the center, as shown in figure 1(a). In this figure, the absorption regions are colored in white with  $\sigma_t = \sigma_a = 10$ ; the purely scattering regions are colored in black and red with  $\sigma_t = \sigma_s = 1$ ; and the red region has an external isotropic source  $S = 1$ . The initial condition for this problem is  $f^m(x, y, \Omega) = 0$ , and the value at the boundary is set to zero.

Figure 1(b) shows a heat map of the particle concentration  $\langle f \rangle$  (in  $\log_{10}$  scale, with nonpositive values marked in white) at time  $t = 2.5$ . To generate these results, we use a degree  $N = 19$  spherical harmonics approximation, a quadrature rule for flux evaluations with precision  $2N + 1$  (that is, it can integrate polynomials on the sphere up to degree  $2N + 1$ , but there is a polynomial of degree  $2N + 2$  that is not integrated precisely), and a filtering parameter  $\sigma_f = 50$ . In space we use a  $256 \times 256$  uniform square mesh, and in time we use a step size of  $\Delta t = 0.0025$ .



### 3. Methodology: incremental batch transfer learning

Most machine learning approaches assume that the training and test data belong to the same feature space and have the same distribution. However, when the feature space or distribution changes (for example, with data from different domains), the models need to be retrained from scratch by using the new training data. For the scientific simulation data studied in this work, the feature space and the data distribution are expected to change, for example, between the climate and nuclear reactor simulation data. Also, since we employ DL models that have millions of tunable parameters, they require large amounts of data in order to build accurate models. We intuitively know that although CAR is performed on data from two different scientific domains, it shares the common task of removing compression artifacts produced by JPEG compression. Moreover, we know that both simulations are hyperbolic PDE-driven. Hence, the learning process would be more efficient and potentially lead to more accurate models if we are able to transfer knowledge between the datasets. Several approaches that facilitate this knowledge transfer are studied under the umbrella of transfer learning [51].

Formally, let a domain be defined as  $\mathcal{D} = \{\mathcal{X}, P(X)\}$ , where  $\mathcal{X}$  is the feature space,  $X$  is the random variable representing the inputs such that  $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}$ , and  $P(X)$  is the marginal probability distribution. A task is defined as  $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$ , where  $\mathcal{Y}$  is the output feature space and  $P(Y|X)$  is the conditional distribution of  $Y$  given  $X$  that defines the mapping between the inputs and outputs learned from the labeled data  $\{x_i, y_i\}$ , where  $x_i \in \mathcal{X}$  and  $y_i \in \mathcal{Y}$ . In traditional machine learning, one assumes that  $\mathcal{D}^s = \mathcal{D}^t$  and  $\mathcal{T}^s = \mathcal{T}^t$ , where the superscript  $s$  represents the source and  $t$  the target. For CAR in this work, we assume that the tasks remain the same ( $\mathcal{T}^s = \mathcal{T}^t$ ) and the difference between datasets is caused only by domain divergence ( $\mathcal{D}^s \neq \mathcal{D}^t$ ). This class of problems is studied under a subcategory of transfer learning called DA [52]. The domain divergence between the datasets could be due to feature spaces being different ( $\mathcal{X}^s \neq \mathcal{X}^t$ ), data distributions being different ( $P(X)^s \neq P(X)^t$ ), or both. Transfer learning between two scientific domains might have to account for change in both feature space and data distribution, while transfer learning in the same scientific domain might need to account only for the change in data distribution.

We adopt the *discrepancy-based deep DA approach* [31] for transfer learning of CAR within and across scientific domains. In this approach the base neural network is trained by using the source data, followed by which the parameters in the first few layers of this network either are frozen or are used as the initial starting point, while the rest of the layers are initialized randomly and trained by using the target data. The assumption in this approach is that fine-tuning the deep network model with labeled target data can diminish the shift between the two domains.

#### 3.1. Incremental batch transfer learning

In this work, we are interested in *in situ* (as the simulation progresses) training, instead of waiting till the end to gather all the data and train the model. To accomplish this by batch learning, we incrementally update the model at every checkpoint. However, we expect to see nonstationarity in the sequence of frames observed as

the simulation progresses and is observed at progressive checkpoints. For this reason, we propose to adopt discrepancy-based deep DA along with experience replay buffers [32–35] that stores a random subset of data collected at each of the previous checkpoints and uses them to augment the training data. Experience replay is a simple yet effective biologically inspired mechanism that can mitigate catastrophic forgetting; in our case we seek to maintain the model's ability to generalize across the progressive checkpoints for the full work. In addition, we synchronize the training of the CAR model with the simulation so that long periods of time do not need to be spent in training the new CAR model at the end of a simulation.

We consider two scenarios for *incremental batch transfer learning*. First is when the training and the prediction data are generated by using simulations that are from the same scientific domain but differ in a parameter value used in the simulation that generated it. Figure 3 depicts this scenario in which the model could start either with a DL-based CAR model pretrained on the simulation data from same scientific domain (that has been trained with historical data) or with randomly initialized weights. Then, at each checkpoint, the model is updated by using the proposed incremental batch transfer learning approach. For this case, the data could be augmented from the historical data or with the data observed at the earlier checkpoints. In the second scenario, we seek to transfer the knowledge between simulations from different scientific domains (see figure 5). Here, we use the model pretrained on one scientific domain as a starting point to do incremental batch transfer learning for a simulation from a different domain. In this case, the data is augmented only from the previous checkpoints.

In addition to the compressed sensing, we compare the proposed *incremental batch transfer learning* approach with an offline transfer learning approach, where the CAR model is trained by using the labeled data obtained at the end of a simulation from a different scientific domain while ignoring the domain divergence between the two datasets. We refer to this as offline because the transfer learning is done after the end of the second simulation when we have access to the full data. The advantage of this approach is that the model does not need to be retrained every time simulation data from a different domain is obtained.

## 4. Methodology: compression artifact removal

In this section we first provide details about the compressed sensing approach used as a baseline and then introduce the DL-based approaches used for image enhancement.

### 4.1. Compressed sensing

Let  $f: [0, 1]^2 \rightarrow \mathbb{R}$  be a function with discrete cosine transform (DCT),

$$\hat{f}_{k_x, k_y} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(x_i, x_j) \cos\left(\frac{i\pi}{N}(k_x + 1/2)\right) \cos\left(\frac{j\pi}{N}(k_y + 1/2)\right), \quad (1)$$

for  $x_i = \frac{i}{N}$ , and  $i, j, k_x, k_y = 0, \dots, N-1$ . Suppose we are given a finite set of DCT coefficients,

$$\hat{\mathbf{f}}_{\Omega} = \{\hat{f}_{k_x, k_y} : (k_x, k_y) \in \Omega\},$$

corresponding to an index set

$$\Omega \subseteq [0, \dots, N-1]^2, \quad |\Omega| = m.$$

The problem of image compression is to reconstruct  $f$  given a subset of DCT coefficients,  $\hat{\mathbf{f}}_{\Omega}$ . The compressed sensing approach is the solution of the optimization problem

$$\min_{\mathbf{g}} \|\Phi \mathbf{g}\|_1 \text{ subject to } \|PD\mathbf{g} - \hat{\mathbf{f}}\|_2 \leq \sigma. \quad (2)$$

Here  $P \in \mathbb{R}^{m \times N^2}$  is a row selector matrix consisting of the subset of rows of the identity matrix corresponding to the index set of the acquired DCT samples  $\Omega$ ,  $D$  is the discrete cosine transform,  $\Phi$  is a discrete sparsifying operator, and  $\|\cdot\|_1$  is the  $l_1$ -norm. Note that although the inverse problem can be regularized in a number of different ways, in (2) a  $l_1$  term is used to promote the sparsity of  $\hat{\mathbf{f}}_{\Omega}$  in the transform domain corresponding to  $\Phi$ . The choice of sparsifying transform  $\Phi$  depends on the particular application. Typical choices include total variation (TV), discrete wavelet transforms, or higher-order transforms such as developed in [53, 54]. In our simulated examples, we have found that TV is effective. This corresponds to the transform

$$\Phi \mathbf{f} = \begin{bmatrix} \nabla_x \mathbf{f} \\ \nabla_y \mathbf{f} \end{bmatrix}, \quad (3)$$

where  $\nabla_x$  and  $\nabla_y$  are the discrete  $x$  and  $y$  derivatives, respectively, with periodic boundary conditions.

Various algorithms such as in [55–60] have been developed to efficiently solve (2). In this investigation we use a split-Bregman technique, as developed in [60] and [57]. In [60] the authors showed that (2) could be solved by using a Bregman iteration of the sequence of two unconstrained problems of the form

$$\mathbf{f}^{k+1} = \min_{\mathbf{f}} \lambda \left( \|\nabla_x \mathbf{f}\|_1 + \|\nabla_y \mathbf{f}\|_1 \right) + \frac{\mu}{2} \|\mathcal{P}\mathcal{D}\mathbf{f} - \hat{\mathbf{f}}^k\|_2, \quad (4)$$

$$\hat{\mathbf{f}}^{k+1} = \hat{\mathbf{f}}^k + \hat{\mathbf{f}} - \mathcal{P}\mathcal{F}\mathbf{f}^{k+1}, \quad (5)$$

where  $\lambda, \mu > 0$  control the fidelity and regularization, respectively. There exists parameter tuning methods for the split split-Bregman methods, but in general setting  $\frac{1}{\lambda} = \mu\sigma^2$  is a robust choice [61]. Solving (4) is generally computationally challenging, but for the case where the DCT samples are taken at integer frequencies, (4) can be solved accurately at the cost of a few DCTs.

#### 4.2. DL-based image enhancement

In this methodology, the image enhancement is posed as a supervised learning problem in which a DL model is trained to take a corrupted (JPEG compressed image in our case) image as an input and predict the restored image (i.e. image free of JPEG compression artifacts in our case) as an output. We adopt CNNs that are a class of deep, feed-forward artificial neural networks commonly applied to analyzing images. CNNs assume that the underlying images are stationary (i.e. statistics of one part of the image are the same as other) and that a spatially local correlation exists between the image pixels. This is particularly important for modeling the artifacts introduced by lossy compression algorithms such as JPEG, whose effect on the underlying image is not dependent on the spatial location of the underlying image features. The convolutional layer is the core building block of a CNN and consists of filters whose size defines the extent of spatial locality assumed; each filter corresponds to a specific feature or pattern in the image. The convolution operation using these filters in each layer ensures stationarity and thus translational invariance. Several convolutional layers are stacked in such a way that the complex image features are learned hierarchically by composing together the features in previous layers. The CNN architectures are ideal for analyzing images, are more efficient to implement than multilayer perceptron models, and vastly reduce the number of parameters in the network, thus decreasing the memory requirement and training time.

In this work, we adopt two different CNN architectures: *enhanced deep super-resolution network* (EDSR) [21] and *residual dense network* (RDN) [22], which are based on residual network and dense networks, respectively. The residual and dense networks are CNN architecture abstractions that differ in how the convolutional layers in the network are connected; both abstractions have shown state-of-the-art performance for image data. We also compare our approach with the *artifacts reduction convolutional neural networks* (AR-CNN) [62] model that is frequently used as a baseline to compare the JPEG artifact removal performance in the offline learning setting. In addition, we compare our offline transfer learning with the *progressive fusion video super-resolution network* (PFNL) approach [29] that exploits nonlocal spatiotemporal correlations in the images sequences to enhance the images. We removed the upsampling layer to reflect the CAR task, and we used all the hyperparameters consistent with the original paper except the learning rate, which we decreases to  $1 \times 10^{-4}$  to mitigate gradient explosion. This approach can be seen as the best-case scenario for accuracy in the offline learning setting.

##### 4.2.1. Enhanced deep super-resolution network.

EDSR utilizes an architecture that builds on ResNet [63], which incorporates skip connections between residual blocks in a deep network and has been shown to work well for a variety of tasks [64]. In the original ResNet architecture, each residual block consists of a series of convolutional and batch normalization layers followed by a non-linear activation function. EDSR proposes to make this architecture efficient for image-to-image tasks such as super-resolution by removing the usage of unnecessary layers (batch normalization layers) to improve the performance of the ResNet model for super-resolution tasks. In this work, we are interested in CAR where the input and output images have the same resolution, so we removed the upsampling layer from EDSR architecture. In addition, we note that all the hyperparameters, both learning parameters and network architecture are kept consistent with the EDSR paper [21] except the total number of epochs, which are specified in the section 5. Moreover, the loss function (L1 loss) is also kept consistent and used for model training.

#### 4.2.2. Residual dense network.

The RDN exploits hierarchical features from the low-resolution images (compressed images in our case) as a way to enhance the images. The RDN approach uses residual dense blocks (RDB), where each block has all the layers densely connected and fused into a  $1 \times 1$  convolutional layer (local feature fusion). In order to further improve performance, residual learning between RDBs and local residual learning is applied in each RDN such that all the RDBs and convolution layers are combined through dense feature fusion, consisting of global feature fusion and global residual learning. In our experiments, we used three RDBs. For this model, all the hyperparameters except the learning rate, optimizer, and the number of epochs are kept consistent with the RDN paper [22]. The learning rate was chosen to be 0.0001, the optimizer was chosen to be Adam, and the number of epochs is same as that used for EDSR experiments. The loss function (L1 loss) is also kept consistent and used for model training.

## 5. Results and discussion

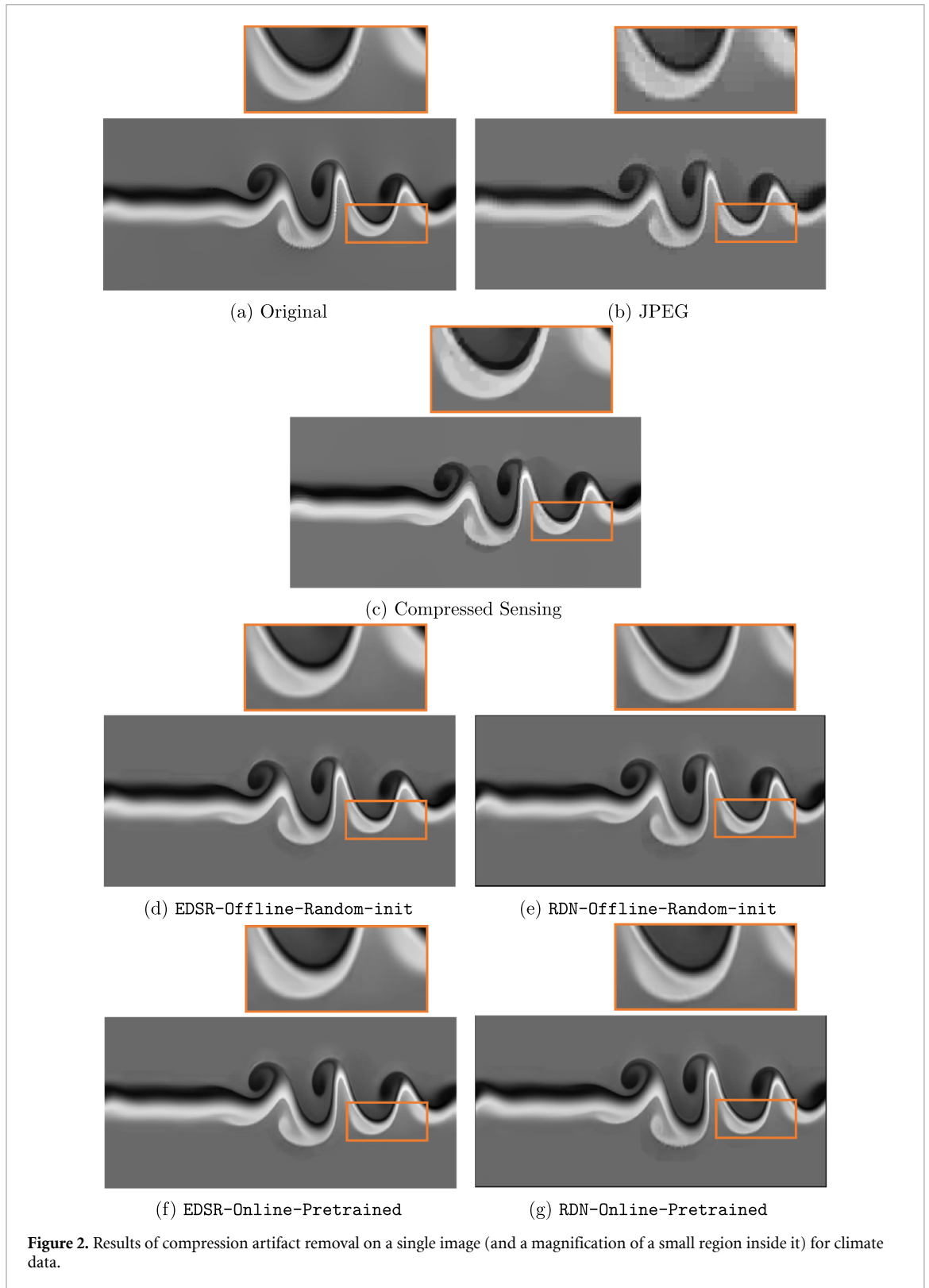
### 5.1. Same domain—climate data

The data used in this experiment is output fields produced from barotropic instability test simulations. The localized bump to the balanced height field ( $\hat{h}$ ) is set to a different value for each simulation and run for a total of 2000 time steps. Every time step produces a 2D scalar field that is then represented as a 2D image, thus producing a series of 2000 images for each simulation. The compressed version of each image is obtained by applying JPEG compression to it at a compression rate of 0.01. Specifically, the value of  $\hat{h}$  is set to 120 m, 180 m, and 240 m in order to generate three different datasets. The value of  $\hat{h} = 120$  m and 240 m represents two extremities of  $\hat{h}$  used in the simulations. The goal of this experiment is to use the data from these extremities to train a DL-based CAR model that can efficiently remove the compression artifacts on JPEG compressed images for data produced at any intermediate  $\hat{h}$  value between 120 m and 240 m. The data corresponding to the initial 700 time steps was not affected by the JPEG compression; hence, they were ignored for all three datasets. The training, testing, and validation splits on the data obtained from simulations with  $\hat{h} = 120$  m and 240 m corresponds to 60%, 20%, and 20%, respectively.

In the EDSR-Offline-Random-init training experiment, the EDSR model (with all weights randomly initialized) is trained by using a total of 1560 image pairs of original and JPEG compressed images on the data obtained from simulations with  $\hat{h} = 120$  m and 240 m. It was then utilized to remove the compression artifacts (prediction) on the 1300 images from simulation data generated with  $\hat{h} = 180$  m. The AR-CNN was trained with the same corresponding training and testing data as the RDN-Offline-Random-init model. The accuracy of this approach was evaluated by comparison with the original uncompressed images. Three metrics—normalized mean squared error (NMSE), structural similarity (SSIM), and peak signal-to-noise ratio (PSNR)—were used to evaluate this accuracy. Comparison also made with the original images for the JPEG compressed images and the images enhanced using Compressed Sensing with 400 iterations. In addition, we compared with the PFNL approach in this offline learning scenario. Since PFNL uses the data sequences that are temporally aligned, however, the random data splits mentioned earlier cannot be directly used. Hence, we use all the 2600 images from simulations with 120 m and 240 m for training but use the same 1300 testing data from the simulation data generated with  $\hat{h} = 180$  m. The results of these comparisons are shown in table 1. The EDSR-Offline-Random-init and RDN-Offline-Random-init show significant accuracy improvement over the AR-CNN, JPEG-Compressed and Compressed Sensing approaches, where the latter approach in this case leads to worsening of the compression artifacts instead of removing them, as indicated by its accuracy metrics that are worse than those of the JPEG-Compressed images. In addition, we observe that their accuracy is on a par with that of PFNL, which indicates that the temporal correlation consideration might not be significant in this context.

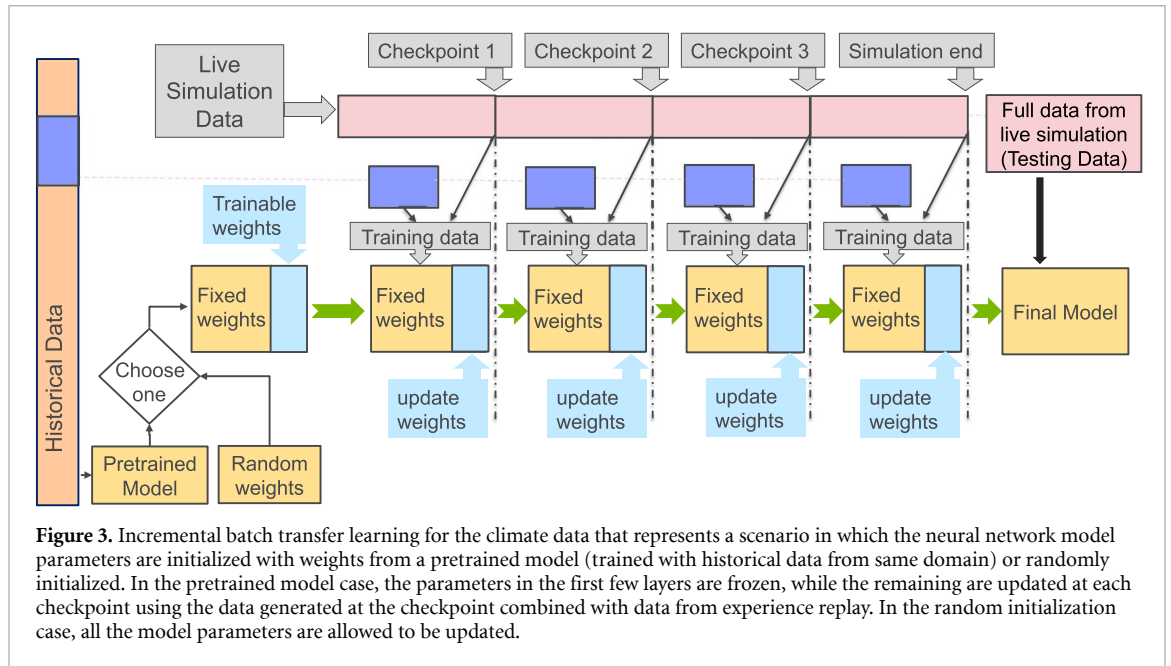
In the EDSR-Online-Pretrained experiment we follow the procedure described in section 3 and shown in figure 3, where the weights of the model trained by using the 1560 images from the training data are used as the pretrained model. The weights in the pretrained model corresponding to the first thirteen residual blocks are frozen, while those in the remaining three blocks are allowed to be updated. At the end of the first checkpoint from the simulation with  $\hat{h} = 180$  m (which is treated as a live simulation), the data generated until this point (325 images) is combined with a random subset of the data corresponding to  $\hat{h} = 120$  m and 240 m (650 images from the 1560 training images) that is added to the replay buffer and used to train the model, that is, to update the trainable weights. This training process is repeated at the end of each checkpoint, with the training data composed of the 325 images generated since the last checkpoint augmented with the same 650 images as used previously. At the end of the simulation and after the final model training, we have a transfer-trained DL model that has the potential to perform a better CAR on the simulation data with  $\hat{h} = 180$  m than by just using the pretrained model on simulations with  $\hat{h} = 120$  m and 240 m. However, the results in table 1 show that there is not a lot of difference between the predictive





accuracies of the EDSR-Offline-Random-init and EDSR-Online-Pretrained models, which indicates that there is not a lot of difference in the underlying features and the corresponding compression artifacts between the simulation data corresponding to  $\hat{h} = 120$  m, 240 m and that with  $\hat{h} = 180$  m. Results similar to EDSR are obtained by using corresponding RDN models (RDN-Online-Pretrained, RDN-offline-Random-init).

Although the compressed sensing approach is unsupervised (i.e. does not require supervised training), its inference time is significantly higher than that of the DL approach. The compressed sensing approach with 400 iterations takes approximately 5 min per image. On the other hand, EDSR-Offline-Random-init and



**Table 1.** Comparison of the mean (standard deviation) predictive accuracy with respect to the original uncompressed data, where the EDSR and RDN models are trained and tested on the climate data.

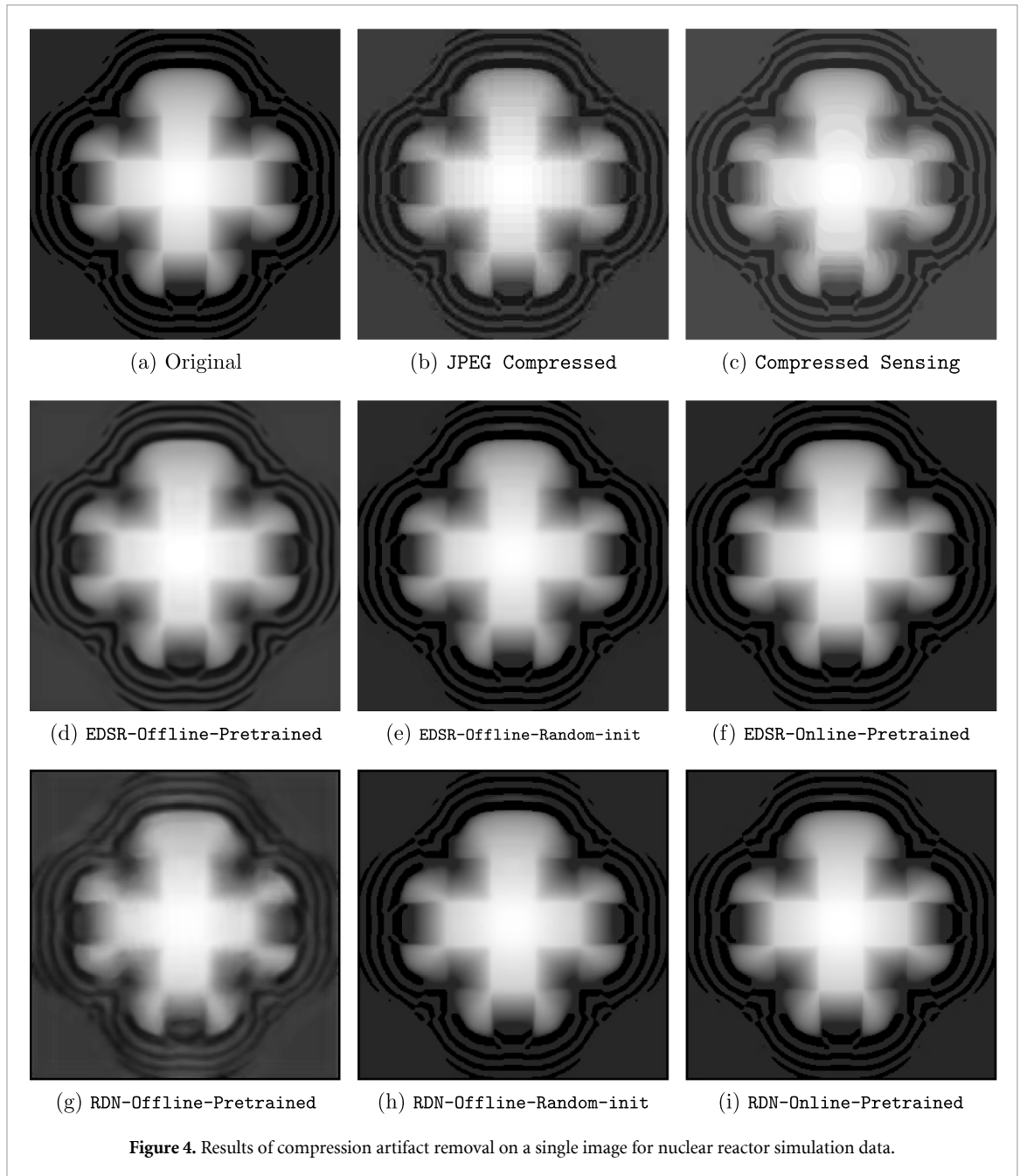
Method	NMSE	SSIM	PSNR
JPEG-Compressed	0.038 (0.027)	0.971 (0.025)	37.245 (6.028)
Compressed Sensing	0.045 (0.021)	0.973 (0.019)	34.534 (3.369)
AR-CNN	0.027 (0.018)	0.989 (0.011)	39.769 (5.103)
PFNL	0.022 (0.017)	0.989 (0.010)	42.381 (6.652)
EDSR-Offline-Random-init	0.029 (0.017)	0.989 (0.011)	42.361 (6.990)
EDSR-Online-Pretrained	0.027 (0.017)	0.989 (0.011)	42.262 (6.760)
RDN-offline-Random-init	0.021 (0.015)	0.990 (0.010)	42.300 (5.472)
RDN-online-Pretrained	0.021 (0.014)	0.992 (0.007)	41.854 (5.220)

EDSR-Online-Pretrained are supervised learning approaches and hence take significant time for training the model (training of EDSR for 1000 epochs took 5.5 h using 1 GPU (Nvidia P100) to get the pretrained model) but only a few milliseconds for prediction. This training, however, is a one-time effort that could be done before the target simulation is run and saved; this time could be further reduced by proposed parallel training approaches. For the incremental batch transfer learning scenario, the training at every checkpoint could be done simultaneously with the live simulation before the next checkpoint data is observed, thus mitigating additional time delay.

## 5.2. Different domain—nuclear reactor simulation

In this experiment we follow the procedure in figure 5. The data corresponds to the 1000 timesteps from the high-resolution nuclear reactor simulation experiment, as depicted in figure 4. We note that the image snapshots (e.g. figure 1(b)) have been converted to grayscale before using them to train the compression artifacts removal model. Three experiments are conducted for this scenario to demonstrate the effectiveness of the proposed incremental batch transfer learning approach across data from different domains.

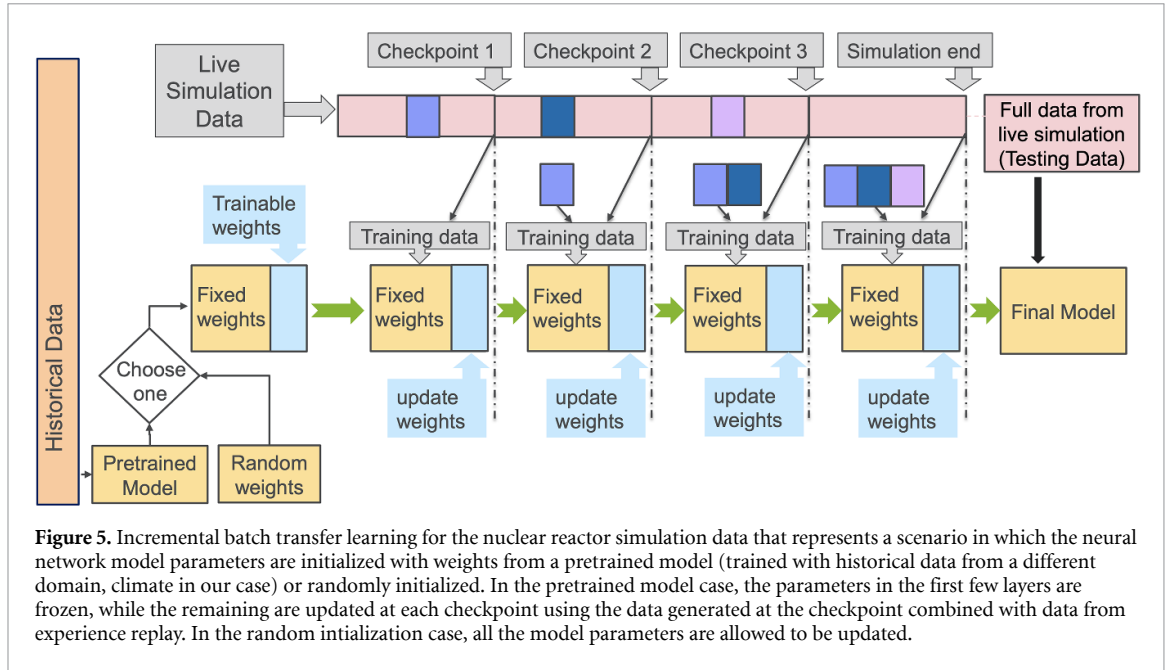
In the first experiment (EDSR-Offline-Pretrained), the model trained by using the climate data corresponding to  $\hat{h} = 120$  m, 240 m is directly used for CAR prediction on all the images corresponding to the nuclear reactor transport simulation offline after the end of the simulation. In the second experiment (EDSR-Online-RandomInit), we start with the EDSR model, where all weights in the model are initialized randomly. The model is updated incrementally at every checkpoint with the data generated since the last checkpoint augmented with a memory buffer that consists of a random subset extracted from the data corresponding to each checkpoint period before the current one. The weights of the updated model are used as the initialization for the training at the subsequent checkpoint. For example, in order to train the model after checkpoint 3, the data generated between checkpoints 2 and 3 is augmented with a random subset of data generated before checkpoint 1 and between checkpoints 1 and 2. The weights of the model trained after



checkpoint 2 are used as the initialization for training after checkpoint 3. This process results in a model that is sequentially updated *in situ* during the live simulation.

The third experiment (EDSR-Online-Pretrained) is similar to the *incremental batch transfer learning* approach described for the climate data, with the difference that the data used to obtain the pretrained model and the data obtained from the live simulation are from two different domains (climate and nuclear reactor simulation, respectively). The data used at each checkpoint to update the model is the same as that used in the second experiment. The weights corresponding to the first thirteen residual blocks are frozen as in the the previous case, thus updating the weights only in the last three residual blocks.

In the second and third experiments, each checkpoint sees a total of 250 images while 50 images are augmented from each of the previous checkpoint intervals. The prediction accuracy of the model in each of these experiments is evaluated on 977 images (between time steps 23 and 1000; the first 22 are not used because the JPEG compression did not affect them) from the nuclear reactor simulation experiment. Table 2 shows the accuracy metrics for each of the experiments along with a comparison of the original uncompressed images with JPEG compressed and the images enhanced using compressed sensing with 400 iterations. The accuracy metrics of the images enhanced with EDSR-Offline-Pretrained are worse than those of the baseline JPEG-Compressed images. The accuracy metrics improve significantly with the



**Figure 5.** Incremental batch transfer learning for the nuclear reactor simulation data that represents a scenario in which the neural network model parameters are initialized with weights from a pretrained model (trained with historical data from a different domain, climate in our case) or randomly initialized. In the pretrained model case, the parameters in the first few layers are frozen, while the remaining are updated at each checkpoint using the data generated at the checkpoint combined with data from experience replay. In the random initialization case, all the model parameters are allowed to be updated.

**Table 2.** Comparison of the mean (standard deviation) predictive accuracy with respect to the original uncompressed data, where the EDSR and RDN models are improved by using incremental batch transfer learning with the nuclear reactor simulation data.

Method	NMSE	SSIM	PSNR
JPEG Compressed	0.154 (0.036)	0.903 (0.053)	26.748 (7.734)
Compressed Sensing	0.156 (0.081)	0.911 (0.069)	27.725 (9.466)
EDSR-Offline-Pretrained	0.175 (0.026)	0.873 (0.040)	24.752 (1.884)
EDSR-Online-Random-init	0.046 (0.029)	0.982 (0.011)	37.022 (2.859)
EDSR-Online-Pretrained	0.026 (0.008)	0.988 (0.009)	41.638 (4.868)
RDN-Offline-Pretrained	0.201 (0.019)	0.803 (0.055)	23.575 (1.654)
RDN-Online-Random-init	0.031 (0.008)	0.986 (0.011)	40.207 (3.732)
RDN-Online-Pretrained	0.024 (0.008)	0.993 (0.005)	42.438 (3.505)

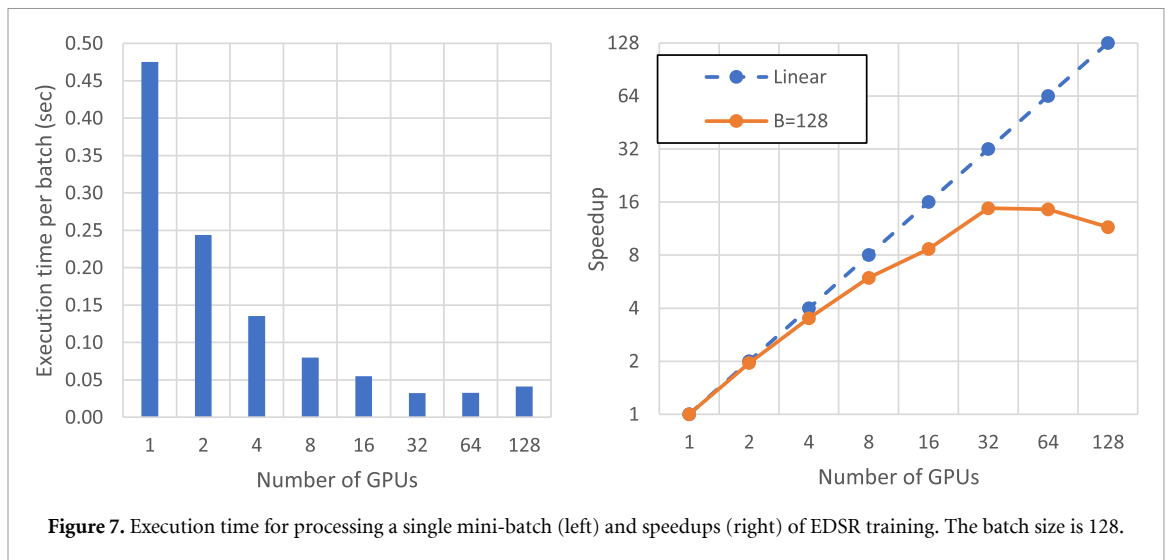
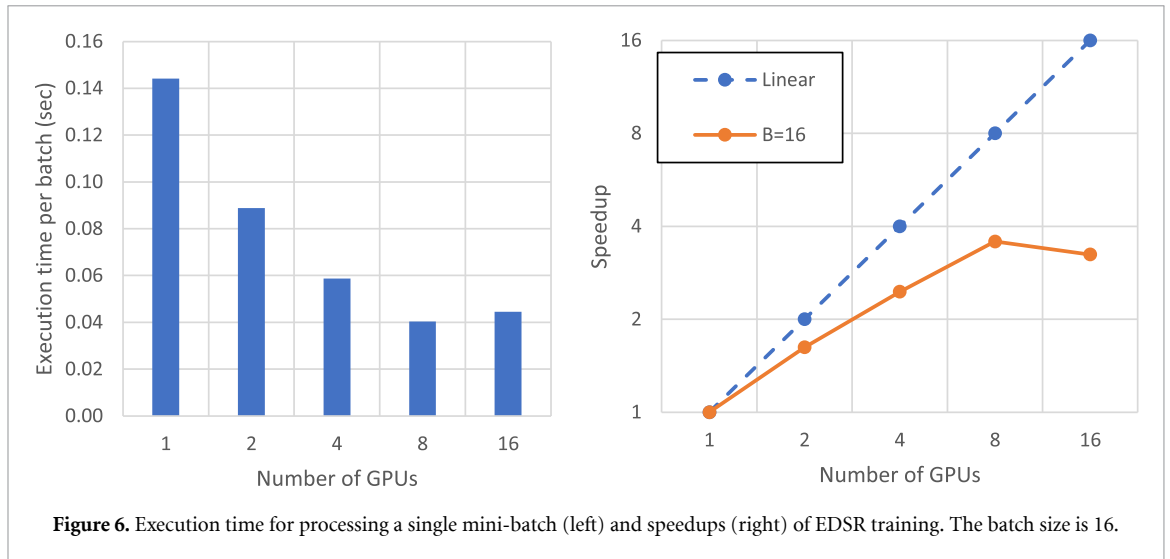
EDSR-Online-Random-init approach where the transfer learning is limited to that across checkpoints from the images corresponding to the same domain of nuclear reactor simulation. We also show that the accuracy metrics are further improved significantly by transfer learning information from the climate data to the nuclear reactor simulation using the EDSR-Online-Pretrained approach. Again, we find that the RDN-based models have behavior similar to those of the corresponding EDSR-based models.

### 5.3. Data-parallel training

Model updating at every checkpoint can also take a long time to complete, hence, we adopt a data parallel training approach to mitigate this problem, and help to synchronize the training of the CAR model with the simulation. Specifically, we adopted a data parallelization approach that exploits the fast convergence rate of synchronous SGD and focuses on addressing the expensive interprocess communication cost by adjusting the mini-batch size and learning rates based on the linear scaling rule. To prevent the convergence rate of SGD from dropping too low, we avoided using a mini-batch size greater than 10% of the training samples; therefore, we set the mini-batch size to the largest power of 2 smaller than 10% of the training samples. In addition, we adjusted the learning rate based on the linear scaling rule proposed in [65] to ensure a good convergence rate while performing the large batch size training. When the mini-batch size increased from  $B$  to  $B'$ , we multiplied the learning rate by  $\frac{B'}{B}$ .

We evaluated the performance of our data-parallel training on Summit, the leadership-class supercomputer at Oak Ridge National Laboratory [66]. Summit consists of 4608 compute nodes, and each compute node contains two sockets of IBM POWER9 CPUs and six NVIDIA Volta V100s GPUs. Our implementation is developed by using TensorFlow [67] (v1.3), which supports GPU acceleration based on CUDA and Horovod [68] (v0.16.0) that provides the parallel training framework.

Figure 6 shows the execution time for training a single mini-batch (left) and the strong-scaling speedups (right) when the mini-batch size  $B$  is 16. We observe a speedup of 3.57 on 16 GPUs, and the speedup curve shows a saturation starting from eight GPUs. This indicates the time when the internode communication



cost starts to overwhelm the computation cost. Note that the maximum degree of parallelism (the maximum number of workers) is 16, because the batch size is 16. In terms of timing for running 1000 epochs, the training time was reduced from 5.21 h on one GPU to 1.61 h on 16 GPUs.

To allow a higher degree of parallelism, we increased the batch size to 128 (the largest power of 2 smaller than 10% of the training samples). We proportionally increased the learning rate to minimize the impact that using a large batch size has on the convergence rate. Figure 7 presents the execution time (left) and the strong-scaling speedups (right) for the large-batch training. We observe a speedup of 14.77 on 128 GPUs, and the speedup curve starts to saturate from 32 GPUs. By increasing the batch size and adjusting the learning rate, we reduced the training time from 2.11 h on one GPU to 0.14 h on 128 GPUs. Note that we achieve a regression accuracy almost the same as that of the small-batch training, even when the mini-batch size is increased (lower than 1% of convergence PSNR difference).

## 6. Conclusion

We developed an *in situ* CAR approach that utilizes CNN-based architectures for feature extraction, discrepancy-based deep DA for transfer learning, and experience replay buffers to mitigate catastrophic forgetting in a unified framework. In this framework, the artifact removal model is updated *in situ* simultaneously as the simulation is run, and it does so efficiently by transferring knowledge from previously trained models using data from within and across domains.

We demonstrate superior recovery of compressed information as compared with standard image compression and traditional optimization-based compressed sensing approaches using lossy JPEG-compressed scientific simulation data from two diverse scientific domains with similar underlying

physics that are driven by hyperbolic PDEs. We demonstrate that online training boosts the accuracy by taking advantage of learning the restricted dynamics that are intrinsic to solutions of hyperbolic PDEs.

The proposed *in situ* approach synchronizes the training of the artifact removal model with the scientific simulation run, in addition to obtaining improved accuracy. This approach saves significant time (as compared with offline model training) and reduces the amount of data that needs to be stored. We further improve the training efficiency using data-parallel training strategies to adjust the mini-batch size and learning rates based on a linear scaling rule.

Identifying similarities between ML methods and PDEs [69], as well as incorporating ML into PDE solvers is an active area of research [70, 71]. A new area of research that could benefit from the frameworks developed in previous research is the adaptation of these methods to enhance online training of streaming data from PDE simulations.

## Acknowledgments

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, through the the FASTMath Institute under Lawrence Livermore National Laboratory contract DE-AC52-07NA27344 and RAPIDS Institute under Argonne National Laboratory contract DE-AC02-06CH11357, both of which are part of the Scientific Discovery through Advanced Computing (SciDAC) program. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725 and computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory. One or more of the co-authors are contractors of the U.S. Government, hence the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

## Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

## ORCID iD

Sandeep Madireddy  <https://orcid.org/0000-0002-0437-8655>

## References

- [1] Jain A K 1989 *Fundamentals of Digital Image Processing* (Upper Saddle River, NJ: Prentice-Hall, Inc.)
- [2] Ramakrishnan L, Fox G and Jha S 2016 STREAM2016: streaming requirements, experience, applications and middleware workshop *Technical Report LBNL-1006355* Lawrence Berkeley National Laboratory (<https://escholarship.org/uc/item/96z57859>)
- [3] Ballard G, Klinvex A and Kolda T G 2019 (arXiv:1901.06043)
- [4] Sayood K 2018 *Introduction to Data Compression* (San Francisco, CA: Morgan Kaufmann Publishers Inc)
- [5] Tian J, Di S, Zhang C, Liang X, Jin S, Cheng D, Tao D and Cappello F 2020 WaveSZ: A hardware-algorithm co-design of efficient lossy compression for scientific data *Proc. 25th ACM Symp. on Principles and Practice of Parallel Programming PPoPP '20* (New York, NY, USA: Association for Computing Machinery) pp 74–88
- [6] Pennebaker W B and Mitchell J L 1992 *Jpeg: Still Image Data Compression Standard* (Springer Science & Business Media)
- [7] Ahmed N, Natarajan T and Rao K R 1974 *IEEE Trans. Comput.* **C-23** 90–3
- [8] Patchett J and Ahrens J 2018 *IEEE Comput. Graph. Appl.* **38** 119–27
- [9] Zhang X, Xiong R, Fan X, Ma S and Gao W 2013 *IEEE Trans. Image Process.* **22** 4613–26
- [10] Minami S and Zakhor A 1995 *IEEE Trans. Circuits Syst. Video Technol.* **5** 74–82
- [11] Chang H, Ng M K and Zeng T 2013 *IEEE Trans. Signal Process.* **62** 718–28
- [12] Candès E J, Romberg J and Tao T 2006 *IEEE Trans. Inf. Theory* **52** 489–509
- [13] Candès E J, Romberg J K and Tao T 2006 *Commun. Pure and Appl. Math.* **59** 1207–23
- [14] Candès E J and Tao T 2006 *IEEE Trans. Inf. Theory* **52** 5406–25
- [15] Donoho D L 2006 *IEEE Trans. Inf. Theory* **52** 1289–306
- [16] Rudin L I, Osher S and Fatemi E 1992 *Physica D* **60** 259–68
- [17] Candès E J, Wakin M B and Boyd S P 2008 *J. Fourier Anal. Appl.* **14** 877–905
- [18] Gunzburger M D, Webster C G and Zhang G 2014 *Acta Numer.* **23** 521–650
- [19] Chkifa A, Dexter N, Tran H and Webster C 2018 *Math. Comput.* **87** 1415–50
- [20] Wu C and Tai X 2010 *SIAM J. Imaging Sci.* **3** 300–39
- [21] Lim B, Son S, Kim H, Nah S and Mu Lee K 2017 Enhanced deep residual networks for single image super-resolution *Proc. Conf. on Computer Vision and Pattern Recognition Workshops* pp 136–44
- [22] Zhang Y, Tian Y, Kong Y, Zhong B and Fu Y 2018 Residual dense network for image super-resolution *Proc. Conf. on Computer Vision and Pattern Recognition* pp 2472–81
- [23] Tai Y, Yang J, Liu X and Xu C 2017 Memnet: A persistent memory network for image restoration *Proc. IEEE Int. Conf. on Computer Vision* pp 4539–47
- [24] Zhang K, Zuo W, Chen Y, Meng D and Zhang L 2017 *IEEE Trans. Image Process.* **26** 3142–55

- [25] Bigdeli S A, Zwicker M, Favaro P and Jin M 2017 Deep mean-shift priors for image restoration *Advances in Neural Information Processing Systems* pp 763–72
- [26] Galteri L, Seidenari L, Bertini M and Del Bimbo A 2017 Deep generative adversarial compression artifact removal *Proc. IEEE Int. Conf. on Computer Vision* pp 4826–35
- [27] Yang R, Xu M, Liu T, Wang Z and Guan Z 2018 *IEEE Trans. Circuits Syst. Video Technol.* **29** 2039–54
- [28] Yang R, Xu M, Wang Z and Li T 2018 Multi-frame quality enhancement for compressed video *Proc. Conf. on Computer Vision and Pattern Recognition* pp 6664–73
- [29] Yi P, Wang Z, Jiang K, Jiang J and Ma J 2019 Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations *Proc. IEEE Int. Conf. on Computer Vision* pp 3106–15
- [30] Xu Y, Gao L, Tian K, Zhou S and Sun H 2019 Non-local ConvLSTM for video compression artifact reduction *Proc. IEEE Int. Conf. on Computer Vision* pp 7043–52
- [31] Yosinski J, Clune J, Bengio Y and Lipson H 2014 How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems* pp 3320–8
- [32] Robins A 1995 *Connect. Sci.* **7** 123–46
- [33] Aljundi R, Lin M, Goujaud B and Bengio Y 2019 Gradient based sample selection for online continual learning *Advances in Neural Information Processing Systems* pp 11816–25
- [34] Lao Q, Jiang X, Havaei M and Bengio Y 2020 (arXiv:2003.04382)
- [35] Bobu A, Tzeng E, Hoffman J and Darrell T 2018 *Int. Conf. on Learning Representations Workshop*
- [36] Williamson D, Drake J, Hack J, Jakob R and Swarztrauber P 1992 *J. Comput. Phys.* **102** 211–24
- [37] Nair R D and Jablonowski C 2008 *Mon. Weather Rev.* **136** 699–711
- [38] McDonald A and Bates J R 1989 *Mon. Weather Rev.* **117** 130
- [39] Galewsky J, Scott R K and Polvani L M 2004 *Tellus A: Dynamic Meteorology and Oceanography* **56A** 429–40
- [40] Abramowitz M and Stegun I (eds) 1972 Ch 9 *Handbook of Mathematical Functions* (Dover Publications)
- [41] Nair R, Thomas S and Loft R 2005 *Mon. Weather Rev.* **133** 876–88
- [42] Sadourny R 1972 *Mon. Weather Rev.* **100** 136–44
- [43] Lewis E and Miller W 1984 *Computational Methods of Neutron Transport* (New York: Wiley)
- [44] Radice D, Abdikamalov E, Rezzolla L and Ott C D 2013 *J. Comput. Phys.* **242** 648–69
- [45] Frank M, Hauck C and Kuepper K 2016 *Commun. Math. Sci.* **14** 1443–65
- [46] Alldredge G W, Hauck C D and Tits A L 2012 *SIAM J. Sci. Comput.* **34** B361–91
- [47] Garrett C K and Hauck C D 2013 *Transp. Theory Stat. Phys.* **42** 203–35
- [48] Atkinson K 1982 *J. Austral. Math. Soc. Ser. B* **23** 332–47
- [49] Walters W F 1987 Use of the Chebyshev-Legendre quadrature set in discrete-ordinate codes *Technical Report* Los Alamos National Lab., NM (USA)
- [50] Brunner T A 2002 Forms of approximate radiation transport *Technical Report SAND2002-1778* Sandia National Laboratories
- [51] Pan S J and Yang Q et al 2010 *IEEE Trans. Knowl. Data Eng.* **22** 1345–59
- [52] Wang M and Deng W 2018 *Neurocomputing* **312** 135–53
- [53] Archibald , Gelb A and Platte R B 2016 *J. Sci. Comput.* **67** 432–52
- [54] Wasserman G, Archibald R and Gelb A 2015 *J. Sci. Comput.* **65** 533–52
- [55] Becker S, Bobin J and Candès E J 2011 *SIAM J. Imaging Sci.* **4** 1–39
- [56] Combettes P L and Wajs V R 2005 *Multiscale Model. Simul.* **4** 1168–200
- [57] Goldstein T and Osher S 2009 *SIAM J. Imaging Sci.* **2** 323–43
- [58] Kim S J, Koh K, Lustig M, Boyd S and Gorinevsky D 2007 *IEEE J. Sel. Top. Signal Process.* **1** 606–17
- [59] Nowak R D and Wright S J et al 2007 *IEEE J. Sel. Top. Signal Process.* **1** 586–97
- [60] Yin W, Osher S, Goldfarb D and Darbon J 2008 *SIAM J. Imaging Sci.* **1** 143–68
- [61] Getreuer P 2012 *Image Process. Line* **2** 74–95
- [62] Yu K, Dong C, Loy C C and Tang X 2016 (arXiv:1608.02778)
- [63] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. Conf. on Computer Vision and Pattern Recognition* pp 770–8
- [64] Szegedy C, Ioffe S, Vanhoucke V and Alemi A A 2017 Inception-v4, Inception-ResNet and the impact of residual connections on learning AAAI vol 4 p 12
- [65] Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y and He K 2017 (arXiv:1706.02677)
- [66] Vazhkudai S S et al 2018 The design, deployment and evaluation of the CORAL pre-exascale systems *Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis* IEEE Press p 52
- [67] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado G S, Davis A, Dean J, Devin M et al 2016 (arXiv:1603.04467)
- [68] Sergeev A and Balso M D 2018 (arXiv:1802.05799)
- [69] Reshniak V and Webster C 2019 (arXiv:1905.10479)
- [70] Han J, Jentzen A and Weinan E W 2018 *Proc. Natl. Acad. Sci.* **115** 8505–10
- [71] Sirignano J and Spiliopoulos K 2018 *J. Comput. Phys.* **375** 1339–64