Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2021-0088

Vol. **31**, No. 1, pp. 188-223
January 2022

A Deep Learning Modeling Framework to Capture Mixing Patterns in Reactive-Transport Systems

N. V. Jagtap[1], M. K. Mudunuru[2] and K. B. Nakshatrala[3,*]

[1] *Department of Mechanical Engineering, University of Houston, Texas 77204, USA.*
[2] *Atmospheric Sciences & Global Change Division, Pacific Northwest National Laboratory, Richland, Washington 99352, USA.*
[3] *Department of Civil & Environmental Engineering, University of Houston, Houston, Texas 77204, USA.*

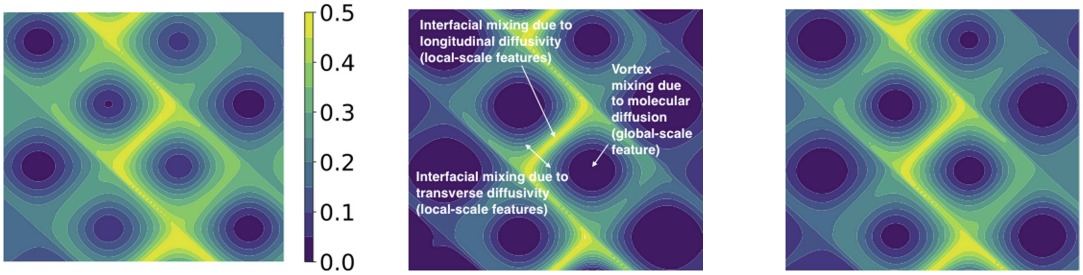
Received 24 April 2021; Accepted (in revised version) 22 July 2021

Abstract. Prediction and control of chemical mixing are vital for many scientific areas such as subsurface reactive transport, climate modeling, combustion, epidemiology, and pharmacology. Due to the complex nature of mixing in heterogeneous and anisotropic media, the mathematical models related to this phenomenon are not analytically tractable. Numerical simulations often provide a viable route to predict chemical mixing accurately. However, contemporary modeling approaches for mixing cannot utilize available spatial-temporal data to improve the accuracy of the future prediction and can be compute-intensive, especially when the spatial domain is large and for long-term temporal predictions. To address this knowledge gap, we will present in this paper a deep learning (DL) modeling framework applied to predict the progress of chemical mixing under fast bimolecular reactions. This framework uses convolutional neural networks (CNN) for capturing spatial patterns and long short-term memory (LSTM) networks for forecasting temporal variations in mixing. By careful design of the framework—placement of non-negative constraint on the weights of the CNN and the selection of activation function, the framework ensures non-negativity of the chemical species at all spatial points and for all times. Our DL-based framework is fast, accurate, and requires minimal data for training. The time needed to obtain a forecast using the model is a fraction ($\approx \mathcal{O}(10^{-6})$) of the time needed to obtain the result using a high-fidelity simulation. To achieve an error of 10% (measured using the infinity norm) for capturing local-scale mixing features such as interfacial mixing, only 24% to 32% of the sequence data for model training is required. To achieve the same level of accuracy for capturing global-scale mixing features, the sequence data required for model training is 64% to 70% of the total spatial-temporal data. Hence, the proposed approach—a fast and accurate way to forecast long-time spatial-temporal mixing patterns in heterogeneous and anisotropic media—will be a valuable tool for modeling reactive-transport in a wide range of applications.

*Corresponding author. *Email address:* knakshatrala@uh.edu (K. B. Nakshatrala)

AMS subject classifications: 35K57, 35-04, 35K51

Key words: Deep learning, reactive-transport, non-negative solutions, spatial-temporal forecasting, pattern recognition, convolutional neural networks (CNN), long short-term memory (LSTM), networks.



(a) Ground truth (b) Prediction: 32% data (c) Prediction: 64% data

The images in this figure compare the ground truth (left) and the predictions (middle and right) from the proposed framework at the end of the simulation (i.e., time = 1.0). The middle image is a forecast based on training using the first 32% of input data; training data is provided until t = 0.32 and the model predicts for the rest of the simulation time. Likewise, the right image provides the forecast based on the 64% of the input data. For capturing local-scale mixing features (e.g., interfacial mixing due to anisotropy), 32% input data will suffice to make predictions within 10% error. In addition, to capture global-scale mixing features (e.g., mixing near vortices caused by molecular diffusion) with similar accuracy, one will need approximately twice the amount of data.

Abbreviations

CNN: Convolutional Neural Network,
DL: Deep Learning,
FEM: Finite Element Method,
LSTM: Long Short-Term Memory,
RNN: Recurrent Neural Network

1 Introduction and motivation

Reactive-transport equations arise in a wide range of scientific domains like epidemiology [53], combustion [41], hypersonic flows [26, 45, 46, 56], subsurface energy [55], contamination remediation [9], air and water quality research [58], and population dynamics [42]. Although the underlying mathematical models in these applications are similar, the associated spatial and temporal scales are much different. For example, the spatial domains in combustion and hypersonic flows are small: an internal-combustion engine for

the former case while the exterior surface of an atmospheric reentry capsule for the latter. In contrast, the spatial domains involved in subsurface remediation and epidemics could range from a few tens to several hundreds of miles. The reaction time scales in combustion are much less than a second while those associated with an epidemic could be very long (months to years). For complex geometries, presence of nonlinearities, and different spatial/temporal scales, analytical treatment of mathematical models arising in the said applications is not viable. Thus, it is customary to resort to numerical simulations.

Finite element [17, 35], finite volume [49], and lattice Boltzmann methods [19] are the contemporary numerical methods to solve reactive-transport equations. But these methods suffer from two major drawbacks. *First*, they do not have an ability to exploit the available data in certain cases to improve prediction accuracy. To elaborate, consider the modeling of an epidemic. During an epidemic, spatial-temporal evolution data gathered by health agencies for the disease is often available [30, 53]. However, the traditional numerical methods cannot utilize such data to improve the accuracy of future predictions; the input to these methods will be limited to data such as the boundary and initial conditions of the solution fields.

The *second* drawback is that reactive-transport simulations using traditional numerical methods are in general compute-intensive for practical problems [8]. For example, numerical solutions of a turbulent combustion problem will require small time-steps for accurate predictions to capture the high-speed traveling reaction front and involved multiple chemical species [27]. On the other hand, for subsurface remediation-related applications or for time-critical scenarios like chemical spills, the evolution of the chemical species for short and long times needs to be understood. Also, the spatial degrees of freedom associated with the subsurface remediation problems could be $\mathcal{O}(10^6)$ to $\mathcal{O}(10^9)$ [14]. It takes hours of simulation time to resolve such spatial and temporal scales adequately even on state-of-the-art supercomputers—despite using thousands of processors, as the calculations in the time domain cannot be parallelized.

The *central aim* of this paper is to present a deep learning (DL) framework that overcomes the said deficiencies of the conventional numerical methods. The efficacy of the framework is demonstrated by applying it to a reactive-transport problem. DL is a subset of machine learning that employs multiple neural layers to extract features successively from the input. An attractive feature of DL-based methods is that they can take advantage of the spatial-temporal data available for a limited portion of the total time of interest. This limited-duration data, obtained from physical experiments or high-fidelity numerical simulations can train a DL model. The resulting trained model can then predict accurate results over the remaining duration of interest much faster than the traditional numerical schemes. Such a model will be extremely useful for real-life situations like a pandemic or simulations requiring multiple runs by varying the simulation parameters of interest to assess their impact on the end results. Once the model is developed, the computational cost for training the model and getting results for the remaining duration of interest will be a fraction of the time required to get the entire solution using the traditional numerical methods. In short, the developed model enables seamless integra-

tion of modeling and data at scale, and significantly reduces the time required to get the solution.

Machine learning has been broadly applied to reactive transport systems and some of the efforts have been described here. [32] developed an on demand machine learning strategy to expedite the equilibrium calculations in reactive transport problems. Using automatic differentiation they calculate sensitivity derivatives at the end of each learning operation. New chemical states at equilibrium are calculated by using a first-order Taylor expansion combined with an equilibrium state that is learned a-priori. This equilibrium state is searched within the database of learned states organized in clusters. Learned equilibrium states are searched within the clusters using a ranking system that is updated during the course of calculation. [15] explored an inverse problem governed by a stochastic non-linear advection-diffusion-reaction equation. With solution samples at a few locations and stochastic diffusivity samples at some locations, they aimed to obtain the full stochastic fields of the concentration and other parameters. They implemented neural networks (NN) using spectral expansions to represent stochasticity to compensate for the sparsity of data. They also presented a method for learning the hyper-parameters of this composite NN. [33] utilized an encoder-decoder based convolutional neural network (CNN) to predict the concentration field. A variety of simulation parameters, boundary conditions and geometries were considered as the input features of the model. The trained CNN model manages to learn the time-dependent behavior of the reaction-diffusion system through the input time feature. The encoders and decoders used by them defined using CNN need to be trained using FEM inputs and solutions. Although their model predicts concentration at a given time accurately, the encoder-decoder training time is extremely large compared to the FEM simulation time. As such, the current ML applications to reactive-transport systems do not address the two deficiencies described above. To the best of our knowledge our paper is the first application of deep learning methods to overcome the said challenges.

The two main DL architectures used in the current framework are *convolutional neural networks* (CNNs) and *long short-term memory* (LSTM) networks. CNNs have existed since the 1980s [47] and are particularly suitable for processing data that has a grid-like topology; e.g., images that are in-fact a 2-dimensional grid of pixels. Their popularity and application have increased tremendously in the last decade due to exponential growth in computing speed as well as volume of data. CNNs offer distinct advantages over the dense/fully connected neural layers that CNNs learn the local patterns in their input features spaces whereas the dense neural layers can only learn the global features. The multiple layers of a CNN can learn the hierarchical patterns. For instance, the first convolution neural layer may learn small local patterns. The second convolution layer may learn broader set of abstract patterns comprised of the features from the first layer, and so on [16, 22]. In the current model, CNNs are used for spatial feature extraction and prediction of reactive-mixing patterns.

A LSTM network is a type of recurrent neural network (RNN), and was developed in the late 1990s [47] while researching the solutions for the well-known vanishing gra-

dient problem in deep neural networks. CNNs are well-suited for processing a grid-like topology while RNNs are ideal for processing a sequence of values. LSTMs preserve the information across many time-steps by preventing the older signals from gradually vanishing during passing of information across the neural network layers [16, 22]. In the current model, the LSTM networks are used for temporal feature extraction and prediction. Thus, CNNs and LSTMs together are used for predicting spatial and temporal evolution of reactive-mixing.

The proposed DL-based framework is validated by applying it to a fast bimolecular reaction problem. High-fidelity simulation results were obtained using finite element method for the entire time domain of interest. The DL-based model is trained using only portion of the total simulation time. The trained model is then used to predict subsequent spatial-temporal evolution of the chemical species. The evolution of the chemical species predicted by this model is compared with the simulation results to assess the accuracy of the model. Sensitivity analysis is performed with respect to amount of training data and various DL algorithm parameters to finalize the hyperparameter values for generating results.

The novel features of the proposed framework are:

(a) *data incorporation*: utilizes the available spatio-temporal species data for accurate prediction of evolution of the species;

(b) *fast*: computationally attractive because it requires smaller time-to-solution compared to high-fidelity simulations;

(c) *non-negative*: ensures non-negative concentration fields at all times and at all spatial points unlike many traditional numerical schemes [36, 39];

(d) *accurate*: provides good accuracy, comparable to high-fidelity numerical solutions;

(e) *predictive*: captures a wide range of reactive mixing; preferential mixing that manifests under small-scale velocity features, as well as incomplete mixing that manifests under large-scale velocity features;

(f) *requires minimal training data*: requires a small amount of training data compared to contemporary data-driven methods; and

(g) *extendable*: can be incorporated into existing simulators with minimal changes to the computer code—an attractive feature for application scientists.

Bereft of such a computational framework, forecasting for reactive-transport applications with large spatial/temporal domains and involving chaotic mixing will be prohibitively time-consuming, and addressing subsurface remediation problems under uncertainties would remain elusive. This paper serves as a proof-of-concept by showing the efficacy of the approach using a two component system; however, our approach is applicable even to those problems involving many chemical species.

An outline of the rest of this article is as follows. We will consider bimolecular reactions as the prototypical reactive-transport model and present the associated governing equations (Section 2). The details about the proposed deep learning based modeling framework for transient reactive-transport will follow this presentation (Section 3). The predictive capabilities and computational performance of the proposed framework will be illustrated using representative numerical examples (Section 4). The paper will end with concluding remarks along with a discussion on potential extensions of this work (Section 5).

2 Governing equations: Reactive-transport

Let Ω be a bounded domain with piecewise smooth boundary $\partial\Omega = \overline{\Omega} - \Omega$, where a superposed bar denotes the set closure. A spatial point is denoted by $\mathbf{x} \in \overline{\Omega}$, and the spatial gradient and divergence operators are denoted by grad$[\cdot]$ and div$[\cdot]$, respectively. The unit outward normal vector to the boundary is denoted by $\widehat{\mathbf{n}}(\mathbf{x})$. The time is denoted by $t \in [0,\mathcal{I}]$, where $\mathcal{I}$ is the length of the time interval of interest.

Consider a bimolecular reaction of the following form:

$$n_A A + n_B B \longrightarrow n_C C, \tag{2.1}$$

where n_A, n_B and n_C are stoichiometric coefficients; A and B are the reactants; and C is the product of the reaction. We denote the molar concentrations of the chemical species by $c_A(\mathbf{x},t)$, $c_B(\mathbf{x},t)$ and $c_C(\mathbf{x},t)$, respectively.

The boundary is divided into two complementary parts. Γ^{D} is that part of the boundary on which concentrations of the chemical species (i.e., Dirichlet boundary conditions) are prescribed. Γ^{N} is the part of the boundary on which diffusive fluxes of the chemical species (i.e., Neumann boundary conditions) are prescribed. Note that

$$\Gamma^{\mathrm{D}} \cup \Gamma^{\mathrm{N}} = \partial\Omega \quad \text{and} \quad \Gamma^{\mathrm{D}} \cap \Gamma^{\mathrm{N}} = \emptyset. \tag{2.2}$$

In the absence of non-reactive volumetric sources and advection, the equations that govern the fate of the reactants and the product in a bimolecular reaction can be written as follows:

$$\frac{\partial c_A}{\partial t} - \mathrm{div}[\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_A]] = -n_A r(c_A,c_B) \quad \text{in } \Omega\times]0,\mathcal{I}[, \tag{2.3a}$$

$$\frac{\partial c_B}{\partial t} - \mathrm{div}[\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_B]] = -n_B r(c_A,c_B) \quad \text{in } \Omega\times]0,\mathcal{I}[, \tag{2.3b}$$

$$\frac{\partial c_C}{\partial t} - \mathrm{div}[\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_C]] = +n_C r(c_A,c_B) \quad \text{in } \Omega\times]0,\mathcal{I}[, \tag{2.3c}$$

$$c_i(\mathbf{x},t) = c_i^{\mathrm{p}}(\mathbf{x},t) \quad \text{on } \Gamma^{\mathrm{D}}\times]0,\mathcal{I}[\quad (i=A,B,C), \tag{2.3d}$$

$$-\widehat{\mathbf{n}}(\mathbf{x}) \bullet \mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_i] = h_i^{\mathrm{p}}(\mathbf{x},t) \quad \text{on } \Gamma^{\mathrm{N}}\times]0,\mathcal{I}[\quad (i=A,B,C), \tag{2.3e}$$

$$c_i(\mathbf{x},t=0) = c_i^0(\mathbf{x}) \quad \text{in } \overline{\Omega} \quad (i=A,B,C), \tag{2.3f}$$

where $c_i^0(\mathbf{x})$ is the initial concentration of the i-th chemical species, $\mathbf{D}(\mathbf{x},t)$ is the anisotropic dispersion tensor, and $r(c_A,c_B)$ is the reactive-part of the volumetric source. $c_i^{\mathrm{p}}(\mathbf{x},t)$ is the prescribed (molar) concentration of the i-th chemical species on Γ^{D}. $h_i^{\mathrm{p}}(\mathbf{x},t)$ is the prescribed diffusive flux of the i-th chemical species on Γ^{N}.

2.1 Invariants and fast reactions

One can simplify the solution procedure by introducing invariants, as done in several papers in the literature; for example, see [39]. The name "invariants" is apparent from the fact that pure transport equations govern the fate of these reaction invariants (i.e., without any reaction term) rather than dispersion-reaction equations. For a bimolecular equation, one can find twelve reaction invariants. However, only two of them are (linearly) independent. Herein we will take the following two independent invariants:

$$c_F(\mathbf{x},t) = c_A(\mathbf{x},t) + \frac{n_A}{n_C} c_C(\mathbf{x},t), \tag{2.4}$$

$$c_G(\mathbf{x},t) = c_B(\mathbf{x},t) + \frac{n_B}{n_C} c_C(\mathbf{x},t). \tag{2.5}$$

The fate of these two invariants are governed by the following equations:

$$\frac{\partial c_F}{\partial t} - \mathrm{div}[\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_F]] = 0 \qquad \text{in } \Omega\times]0,\mathcal{I}[, \tag{2.6a}$$

$$c_F(\mathbf{x},t) = c_F^{\mathrm{p}}(\mathbf{x},t) := c_A^{\mathrm{p}}(\mathbf{x},t) + \left(\frac{n_A}{n_C}\right) c_C^{\mathrm{p}}(\mathbf{x},t) \qquad \text{on } \Gamma^{\mathrm{D}}\times]0,\mathcal{I}[, \tag{2.6b}$$

$$(-\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_F])\bullet\widehat{\mathbf{n}}(\mathbf{x}) = h_F^{\mathrm{p}}(\mathbf{x},t) := h_A^{\mathrm{p}}(\mathbf{x},t) + \left(\frac{n_A}{n_C}\right) h_C^{\mathrm{p}}(\mathbf{x},t) \qquad \text{on } \Gamma^{\mathrm{N}}\times]0,\mathcal{I}[, \tag{2.6c}$$

$$c_F(\mathbf{x},t=0) = c_F^0(\mathbf{x}) := c_A^0(\mathbf{x}) + \left(\frac{n_A}{n_C}\right) c_C^0(\mathbf{x}) \qquad \text{in } \overline{\Omega}, \tag{2.6d}$$

and

$$\frac{\partial c_G}{\partial t} - \mathrm{div}[\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_G]] = 0 \qquad \text{in } \Omega\times]0,\mathcal{I}[, \tag{2.7a}$$

$$c_G(\mathbf{x},t) = c_G^{\mathrm{p}}(\mathbf{x},t) := c_B^{\mathrm{p}}(\mathbf{x},t) + \left(\frac{n_B}{n_C}\right) c_C^{\mathrm{p}}(\mathbf{x},t) \qquad \text{on } \Gamma^{\mathrm{D}}\times]0,\mathcal{I}[, \tag{2.7b}$$

$$(-\mathbf{D}(\mathbf{x},t)\,\mathrm{grad}[c_G])\bullet\widehat{\mathbf{n}}(\mathbf{x}) = h_G^{\mathrm{p}}(\mathbf{x},t) := h_B^{\mathrm{p}}(\mathbf{x},t) + \left(\frac{n_B}{n_C}\right) h_C^{\mathrm{p}}(\mathbf{x},t) \qquad \text{on } \Gamma^{\mathrm{N}}\times]0,\mathcal{I}[, \tag{2.7c}$$

$$c_G(\mathbf{x},t=0) = c_G^0(\mathbf{x}) := c_B^0(\mathbf{x}) + \left(\frac{n_B}{n_C}\right) c_C^0(\mathbf{x}) \qquad \text{in } \overline{\Omega}. \tag{2.7d}$$

In this paper, we assume that the bimolecular reaction is fast—the time-scale of the reaction is faster than that of the dispersion process. This assumption implies that at any spatial point and at instance of time, concentration of only one of the reactants A or B

could be non-zero. Said differently, if both c_A and c_B are non-zero, the reaction will proceed further instantaneously until one of the reactants is depleted completely.

For *fast* bimolecular reactions, the solution procedure can be further simplified by noting that reactants A and B cannot coexist. Once $c_F(\mathbf{x},t)$ and $c_G(\mathbf{x},t)$ are known, the concentrations of the reactants and the product are calculated as follows:

$$c_A(\mathbf{x},t) = \max\left[c_F(\mathbf{x},t) - \left(\frac{n_A}{n_B}\right) c_G(\mathbf{x},t), 0\right], \tag{2.8a}$$

$$c_B(\mathbf{x},t) = \left(\frac{n_B}{n_A}\right) \max\left[-c_F(\mathbf{x},t) + \left(\frac{n_A}{n_B}\right) c_G(\mathbf{x},t), 0\right], \tag{2.8b}$$

$$c_C(\mathbf{x},t) = \left(\frac{n_C}{n_A}\right) \left(c_F(\mathbf{x},t) - c_A(\mathbf{x},t)\right). \tag{2.8c}$$

The appendix provides details of the non-negative finite element formulation used to generate the high-fidelity simulation data employed herein. To summarize, the mathematical model considered in this paper makes the following assumptions:

(i) Advection is neglected.

(ii) Non-reactive volumetric source is neglected for all chemical species.

(iii) The dispersion tensor is assumed to be the same for all the chemical species involved in the reaction.

(iv) Only the diffusive part of the flux is prescribed on the boundary.

(v) The time-scale of the reaction is much faster than the time-scale associated with the dispersion process.

Some prior works [7, 34, 52] have used a similar bimolecular reaction model, and utilized unsupervised and supervised machine learning methods to perform data mining in the simulation data. Specifically, these works have discovered hidden features, estimated relative importance of reaction-transport model input parameters, and emulated key quantities of interest (e.g., degree of mixing, product yield, species decay). However, these works did not capture the spatial-temporal mixing patterns, which is the main uniqueness of this paper.

3 Proposed deep learning modeling framework

In this section, we will present our proposed deep learning framework. The details on the neural architecture of our non-negative CNN-LSTM model will be outlined first. We will then describe the training process for ingesting reactive-transport simulation data to make predictions at future time-steps.

3.1 Deep learning model architecture

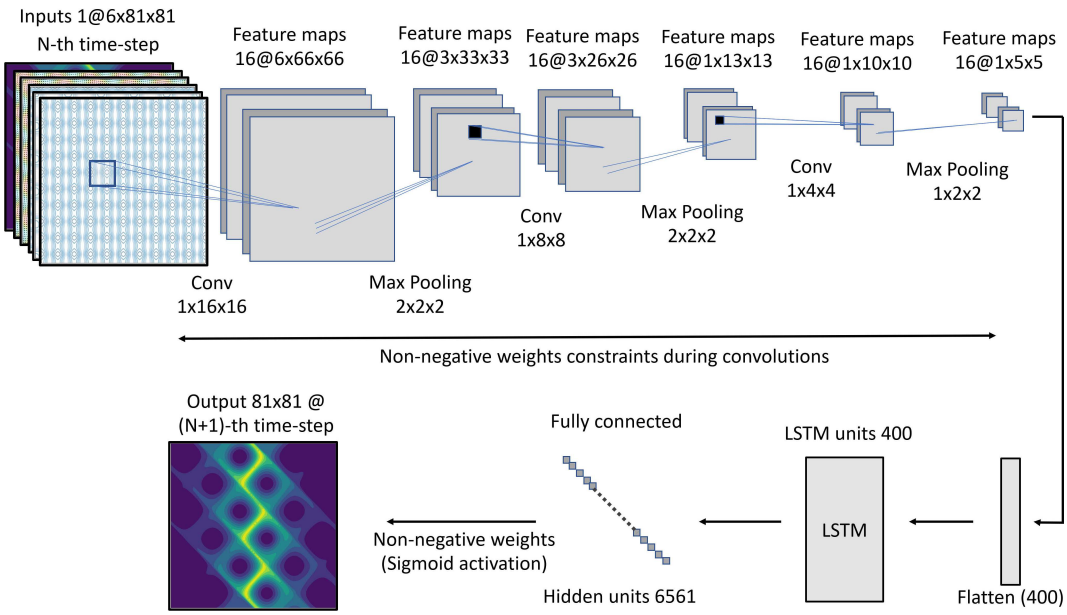
It is well known that many of the traditional finite element formulations for the reactive-transport equations suffer from a deficiency: they could produce negative concentrations [36]. For example, the standard single-field Galerkin formulation produces negative values and spurious node-to-node oscillations for the primary variables in advection- and reaction-dominated diffusion equations. The negative concentrations are unphysical and can erode the solution accuracy in the entire domain. Additionally, if these erroneous numerical solutions are used to train deep learning models, then they could produce inaccurate forecasts. Therefore, ensuring non-negative concentrations in the ground truth and during DL-model training/predictions is essential.

The framework developed herein ensures non-negativity of the predicted concentrations through careful selection of the trainable parameters. Fig. 1 shows a pictorial description of the proposed deep learning model. Fig. 1(a) shows a schematic of the non-negative CNN-LSTM architecture, which is developed to forecast the evolution of spatial patterns in the concentration of product C. The inputs to the convolutional layers are an image stack as shown in Fig. 2. The convolutional layers learn the underlying representations (e.g., feature maps) of the mixing process based on the observed patterns in the concentration, velocity field, and anisotropic dispersion at the previous time-steps. The last dense layers along with the LSTM units predict the evolution of the reactive-mixing based on the learned patterns (e.g., features maps or representations) from convolutional layers. Fig. 1(b) shows a schematic of prediction at future-times based on the trained non-negative CNN-LSTM model. Concentrations (i.e., ground truth data obtained from the non-negative FEM) at initial time-steps along with velocity field and dispersion tensor are used to train the deep learning model. As described in Fig. 2, to forecast the product C concentration at subsequent time-steps, we input these image stacks to the trained non-negative CNN-LSTM model.

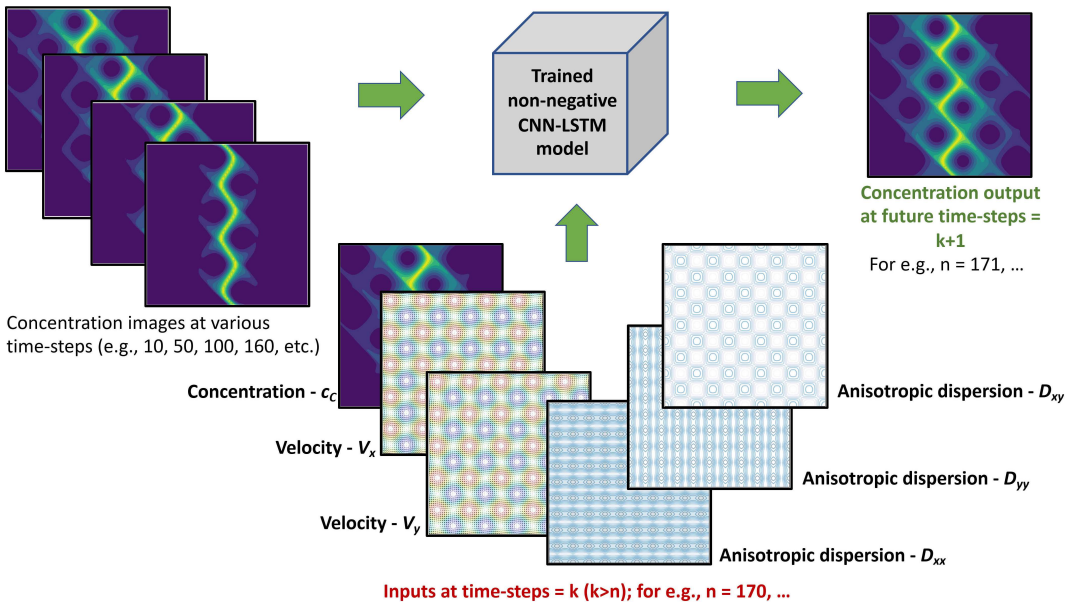
Keras and Tensorflow packages [16] are used to build our CNN-LSTM model (see Fig. 1(a)). The convolutional neural layers in our model are trained to find a set of transformations that turn the input image stack into simpler and more general feature maps that represent the reactive-mixing process. The convolution operator learned by a filter that is applied at every spatial point on the input image stack is a local linear combination of neighboring points of that input. Pooling and activation operations are convex operations. These are applied to the convolution operations, which make our CNN a form of high-dimensional composition of spline operators [11].

Following Goodfellow et al. [22], the CNN layers in the proposed model can be mathematically described as follows: Let us assume L^{r-1} is a $(r-1)$-th neuron layer of size $\mathcal{R}\times\mathcal{R}$ followed by a convolutional layer. A filter F^{r-1} of size $\mathcal{M}\times\mathcal{M}$ is applied to L^{r-1}. Then, our convolutional layer output will be of size $(\mathcal{N}-\mathcal{M}+1)\times(\mathcal{N}-\mathcal{M}+1)$. This convolution operation is given as follows:

$$s_{ij}=\sum_{p=0}^{M-1}\sum_{q=0}^{M-1} f_{pq}\, l^{r-1}_{(i-p)(j-q)}, \quad \text{where} \quad f_{pq}\in F^{r-1}, l^{r-1}_{ij}\in L^{r-1}. \tag{3.1}$$



(a) Schematic of non-negative CNN-LSTM architecture for spatio-temporal sequence forecasting



(b) RT predictions at future time-steps based on a trained non-negative CNN-LSTM model

Figure 1: **Proposed deep learning modeling framework:** This figure shows a pictorial description of the proposed DL-based framework. The top figure shows the non-negative CNN-LSTM architecture and the bottom figure shows the predictions at the future time-steps based on the trained model. Concentration images at time steps 1 to n are used to develop a trained model. The trained model takes input at time step k $(>n)$ and outputs the prediction at time step $k+1$.

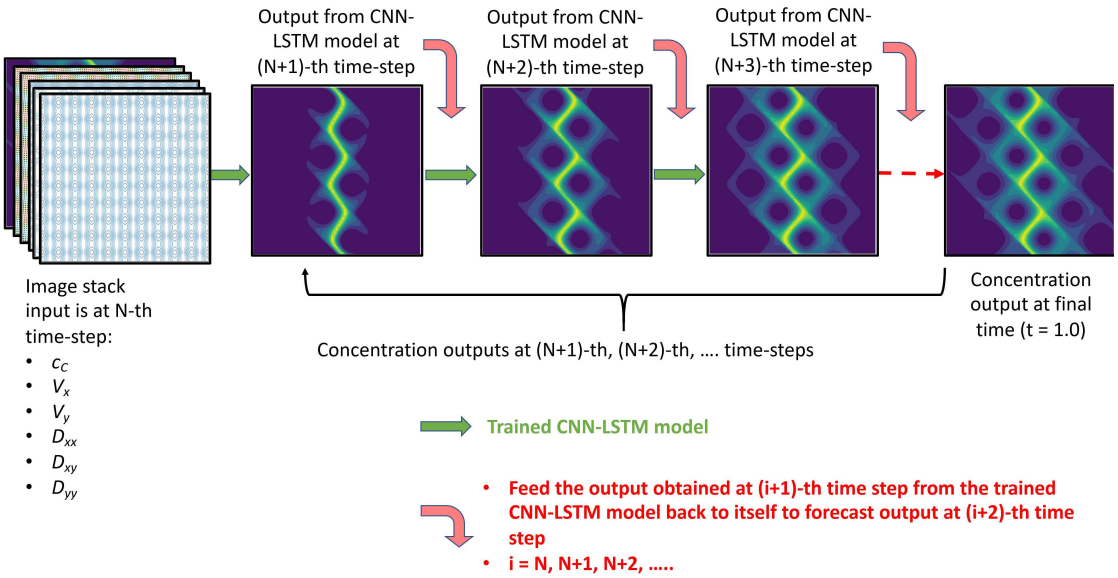


Figure 2: **Summary of forecasting product concentrations:** A pictorial description of spatio-temporal sequence forecasting of the reactive-transport data using the trained non-negative CNN-LSTM model. The model is trained for the first N time-steps and predictions are made from the $N+1$-th time-step till the end of simulation time.

Nonlinearity is then applied to s_{ij} to get the next neuron layer L^r. This operation is given by $l^r_{ij}=\sigma(s_{ij})$, where $l^r_{ij}\in L^r$ and σ is the sigmoid function. There is no learning in the max-pooling layers. The pooling operation simply take certain $k\times k$ region within the $(\mathcal{N}-\mathcal{M}+1)\times(\mathcal{N}-\mathcal{M}+1)$ and outputs a single value. This value is the maximum in that $k\times k$ region. In our case, for a given neuron layer L^r, max-pooling operation will output a $\frac{(\mathcal{N}-\mathcal{M}+1)}{k}\times\frac{(\mathcal{N}-\mathcal{M}+1)}{k}$ layer. This is because each $k\times k$ matrix is reduced to just a single value through the max function.

The LSTM layers are primarily composed of three gates—input gate, forget gate, and output gate. Sigmoid activation functions are used in these gates to give non-negative values at different states. Input gate tells us the new information to be stored in the LSTM cell state. Forget gate tells us the features to be thrown away or irrelevant. The output gate provides the activation required to estimate the final output of the LSTM block at time-step j. Mathematically, the flow of information from these gates can be written as follows:

$$\mathbf{f}_j=\sigma(\mathbf{W}_f\mathbf{z}_j+\mathbf{U}_f\mathbf{h}_{j-1}+\mathbf{b}_f), \tag{3.2a}$$
$$\mathbf{i}_j=\sigma(\mathbf{W}_i\mathbf{z}_j+\mathbf{U}_i\mathbf{h}_{j-1}+\mathbf{b}_i), \tag{3.2b}$$
$$\mathbf{o}_j=\sigma(\mathbf{W}_o\mathbf{z}_j+\mathbf{U}_o\mathbf{h}_{j-1}+\mathbf{b}_o), \tag{3.2c}$$
$$\tilde{\mathbf{a}}_j=\tanh(\mathbf{W}_a\mathbf{z}_j+\mathbf{U}_a\mathbf{h}_{j-1}+\mathbf{b}_a), \tag{3.2d}$$

$$\mathbf{a}_j = \mathbf{f}_j \odot \tilde{\mathbf{a}}_{j-1} + \mathbf{i}_j \odot \tilde{\mathbf{a}}_j, \tag{3.2e}$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{a}_j), \tag{3.2f}$$

where $\odot$ denotes the Hadamard product [25]. At time-step $j=0$, $\mathbf{a}_j = 0$ and $\mathbf{h}_j = 0$. $\mathbf{z}_j$ is the input vector to the LSTM cell. $\mathbf{h}_j$ is the hidden state vector, which is also known as output vector of the LSTM cell. $\mathbf{f}_j$, $\mathbf{i}_j$, and $\mathbf{o}_j$ are the activation vectors for forget, input, and output gates, respectively. $\tilde{\mathbf{a}}_j$ is the LSTM cell input activation vector. $\mathbf{a}_j$ is the LSTM cell state vector. $\mathbf{W}_\alpha$ and $\mathbf{U}_\alpha$, $\forall \alpha = f,i,o,\text{and},a$, are the weight matrices of the input and recurrent connections. $\mathbf{b}_\alpha$ are the corresponding bias vectors. $\mathbf{W}_\alpha$, $\mathbf{U}_\alpha$, and $\mathbf{b}_\alpha$ are learned during the training process.

Fig. 2 shows a pictorial description of the spatial-temporal sequence forecasting (STSF) of the reactive-transport data. STSF is performed using the non-negative CNN-LSTM model that is built on the proposed DL-based model architecture as shown in Fig. 1.

To summarize, our framework ingests data from zero to N-th time-step for training the CNN-LSTM model. The ingested data is an image stack consisting of $c_C(x,y)$, v_x, v_y, D_{xx}, D_{xy}, and D_{yy} images at N-th time-step. $c_C(x,y)$ is the concentration of product C from high-resolution numerical simulation using the non-negative FEM (ground truth). v_x and v_y are the x- and y-components of the vortex-based velocity fields given by Eqs. (4.3)-(4.4). D_{xx}, D_{xy}, and D_{yy} are the xx-, xy-, and yy-components of the anisotropic dispersion tensor given by Eq. (4.1). The trained CNN-LSTM model uses these images to produce a sequence of predictions for $c_C(x,y)$ for time-steps greater than N. We also note that the trained CNN-LSTM model constantly ingests data that it produces after N-th time-step to make forecasts until the end of simulation time.

3.2 Non-negative CNN-LSTM model training

The architecture of our CNN consists of three convolutional neural layers with 16 filters. We initialize filters with the Glorot uniform initializer, which is also called Xavier uniform initializer [21]. The CNN layer weights are constrained to be non-negative and non-linear mapping is based on ReLU activation function [38]. This allows us to learn non-negative feature maps. A max-pooling operation is applied after each convolution. The final CNN layer is flattened and followed by a LSTM layer with 400 units. This LSTM layer output is then fed to a fully connected layer of size 6561 that has a sigmoid activation function. Note that the dense layer outputs the concentration at the next time-step and its weights are also constrained to be non-negative. The output of the dense layer is then reshaped to a 2D concentration image of product C. The entire model is compiled with an Adam optimizer [31] with loss being the mean squared error. The resulting model has a total of 3,298,785 trainable parameters. Training our model is accomplished by backpropagation. The weights in the CNN-LSTM model are updated iteratively on batches of data from the training set. The batch size is 6 and the training is terminated after 100 epochs. The proposed model is updated after seeing each batch by using gradient descent and

backpropagation. The weights are adjusted to minimize the mean squared error of the proposed deep learning model (i.e., through the gradient of the error with respect to the weights). Note that we performed sensitivity analysis with respect to model parameters like LSTM units, batch size and epochs to finalize these hyperparameters used to generate results presented in this paper.

4 Representative numerical results

In this section, we present numerical results to evaluate accuracy and efficiency of the proposed framework. We will first describe an initial boundary value problem that is used to generate data; this problem provides insights into reactive-mixing in subsurface under weakly chaotic flow fields (e.g., vortex-based structures). We then summarize the input parameters used to generate data and train the CNN-LSTM models. Next, we will describe in-detail on the accuracy of the proposed DL-based framework. Ground truth and CNN-LSTM model predictions are compared to show the predictive capability of these models. Finally, we will discuss the computational cost associated with training and testing the deep learning models.

4.1 Reaction tank problem

Fig. 3 provides a pictorial description of the initial boundary value problem. The computational domain is a square with side length $L=1$. Zero flux boundary conditions are enforced on the sides of the domain. The non-reactive volumetric source $f_i(\mathbf{x},t)$ is equal to zero for all the chemical species. Initially species A and species B are segregated such that species A is placed in the left half of the domain while species B is placed in the right half. The stoichiometric coefficients are selected as $n_A=n_B=n_C=1$. The total time of interest is selected as $\mathcal{I}=1$. The dispersion tensor is chosen from the subsurface literature [43] and is given as follows:

$$\mathbf{D}_{\text{subsurface}}(\mathbf{x})=D_m\mathbf{I}+\alpha_T\|\mathbf{v}\|\mathbf{I}+\frac{\alpha_L-\alpha_T}{\|\mathbf{v}\|}\mathbf{v}\otimes\mathbf{v}. \tag{4.1}$$

where D_m is the molecular diffusivity, α_L is the longitudinal diffusivity, α_T is the transverse diffusivity, $\mathbf{I}$ is the identity tensor, $\otimes$ is the tensor product, $\mathbf{v}$ is the velocity vector field, and $\|\bullet\|$ is the Frobenius norm. As discussed in Section 2.1, we have neglected advection. A model velocity field is used to define the anisotropic dispersion tensor through the following stream function [6, 35, 50]:

$$\psi(\mathbf{x},t)=\begin{cases}\frac{1}{2\pi\kappa_f}\left(\sin(2\pi\kappa_f x)-\sin(2\pi\kappa_f y)+v_0\cos(2\pi\kappa_f y)\right) & \text{if } \nu T\le t<\left(\nu+\frac{1}{2}\right)T,\\ \frac{1}{2\pi\kappa_f}\left(\sin(2\pi\kappa_f x)-\sin(2\pi\kappa_f y)-v_0\cos(2\pi\kappa_f x)\right) & \text{if } \left(\nu+\frac{1}{2}\right)T\le t<(\nu+1)T,\end{cases} \tag{4.2}$$

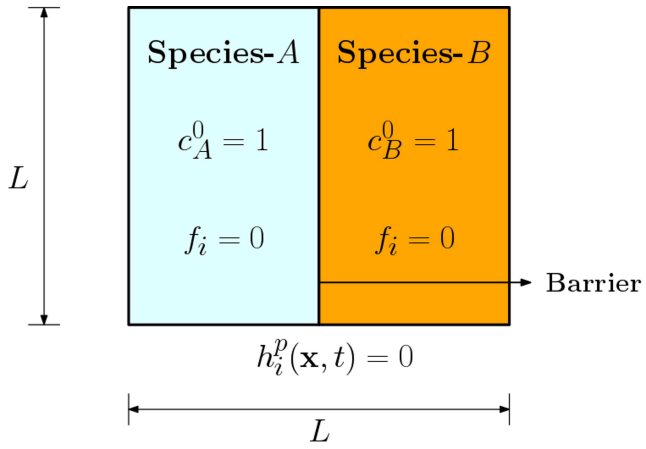


Figure 3: **Problem description:** A pictorial description of the initial boundary value problem. The figure shows the model domain for generating reactive-transport simulation data. The length of the domain 'L' is assumed to be equal to 1. Zero flux boundary conditions are enforced on all sides of the domain ($h_i^{\mathrm{p}}(\boldsymbol{x},t)=0.0$). Focus is on fast irreversible bimolecular reactions, where species A and B are initially separated by a barrier. They are on the left and right sides of the domain, respectively. Initial concentration of A and B are equal to 1.0. The species are allowed to mix and react after $t>0$ according to the velocity field given by Eqs. (4.3)-(4.4). The non-reactive volumetric sources for all the chemical species are assumed to be equal to zero.

where ν=0,1,2,$\cdots$ is an integer. $\kappa_f L$ and T are characteristic scales of the flow field. Using Eq. (4.2), the divergence-free velocity field components are given as follows:

$$\mathrm{v}_x(\mathbf{x},t) = -\frac{\partial \psi}{\partial \mathrm{y}} = \begin{cases} \cos(2\pi\kappa_f y) + v_0 \sin(2\pi\kappa_f y) & \text{if } \nu T \le t < \left(\nu+\frac{1}{2}\right) T, \\ \cos(2\pi\kappa_f y) & \text{if } \left(\nu+\frac{1}{2}\right) T \le t < (\nu+1)\, T, \end{cases} \tag{4.3}$$

$$\mathrm{v}_y(\mathbf{x},t) = +\frac{\partial \psi}{\partial \mathrm{x}} = \begin{cases} \cos(2\pi\kappa_f x) & \text{if } \nu T \le t < \left(\nu+\frac{1}{2}\right) T, \\ \cos(2\pi\kappa_f x) + v_0 \sin(2\pi\kappa_f x) & \text{if } \left(\nu+\frac{1}{2}\right) T \le t < (\nu+1)\, T. \end{cases} \tag{4.4}$$

4.2 Data generation

A non-negative finite element method described in Appendix A is used to generate high-resolution data for training and testing the DL-based framework. Low-order triangular finite element mesh of size 81×81 with 6,561 degrees-of-freedom is employed for discretizing the domain. Backward Euler time-stepping scheme with $\Delta t = 0.001$ is used to advance the simulation time from $t = 0.0$ to $t = 1.0$. Note that in our previous works [35,36,39], we have performed h-convergence and time-step-refinement studies to solve a similar system of equations described in Section 2. The outcome of these studies showed that 81×81 mesh with $\Delta t = 0.001$ was sufficient to achieve accurate numerical solutions to capture fine-scale spatial-temporal variations caused by anisotropic dispersion.

To demonstrate the applicability of the proposed DL-based framework, we have used a total of four high-resolution numerical simulations to train and test the non-negative

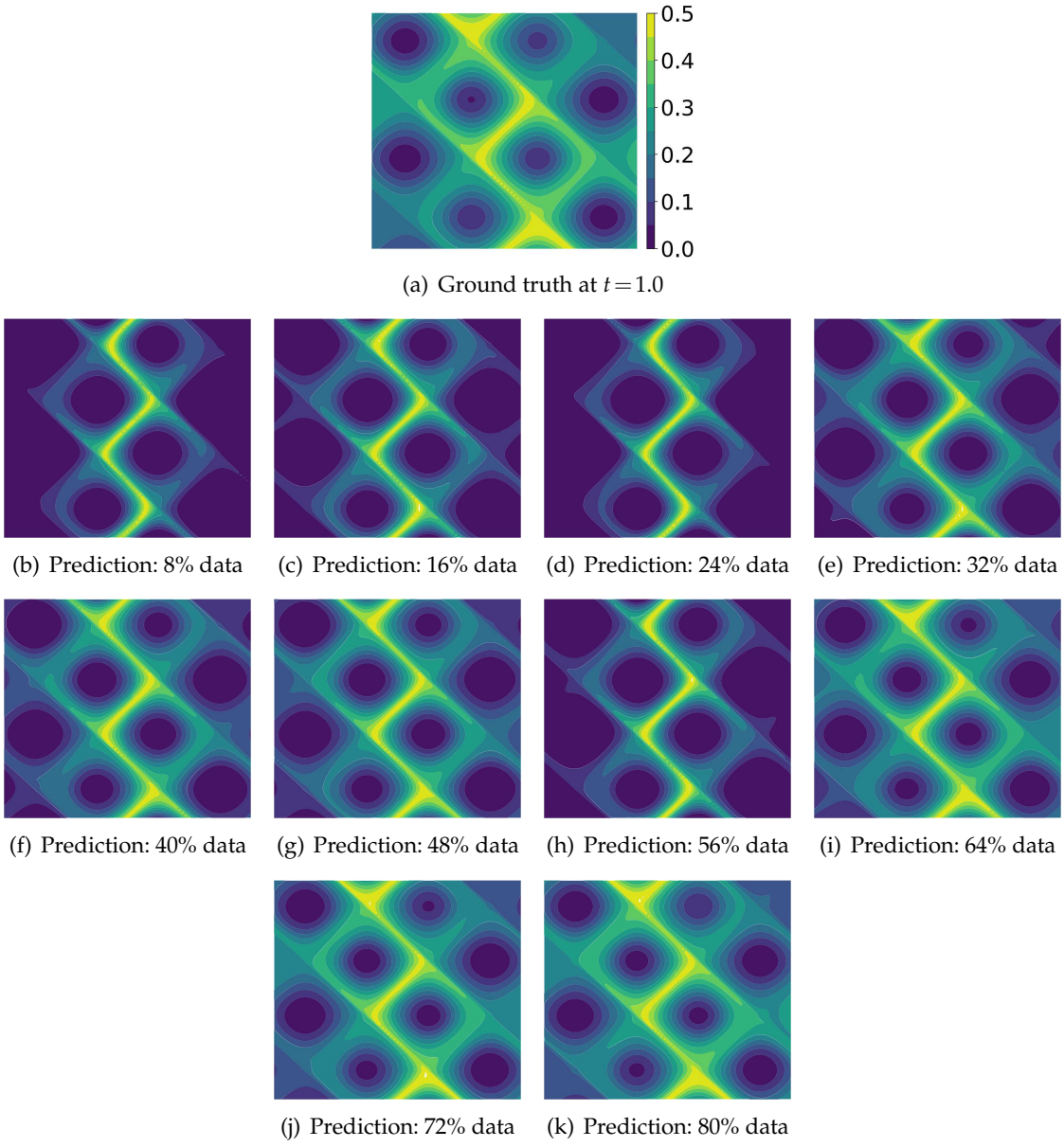
CNN-LSTM models. Each simulation has 1000 images of c_C, whose size is 81×81. A detailed description of the simulation data (a total of 2315 realizations) generated for machine learning analysis is described in References [7,34,52]. Herein, we use a subset of this massive dataset for testing our proposed framework. The reaction-diffusion model input parameters associated with this subset are as follows: $\kappa_f L$ = [2,3,4,5], $v_0 = 10^{-1}$, $\frac{\alpha_L}{\alpha_T} = 10^4$, $D_m = 10^{-3}$, and $T = 10^{-4}$. That is, $\kappa_f L$ is varied and other parameters (e.g., v_0, $\frac{\alpha_L}{\alpha_T}$, D_m, T) are kept constant. The resulting data is shown in Fig. 4(a), Fig. 6(a), Fig. 8(a), Fig. 10(a). The above selected parameters represent reactive-mixing under high-anisotropy. $\frac{\alpha_L}{\alpha_T} = 10^4$ corresponds to a scenario where longitudinal dispersion (α_{L} along the streamlines) is 10^4 times higher than transverse dispersion (α_{T} across streamlines). Higher values of $\kappa_f L$ enhance mixing and lower values results in regions/islands where species rarely mix. $v_0 = 10^{-1}$ results in very small perturbations in the flow field, thereby, preserving the underlying symmetry in the mixing patterns. $D_m = 10^{-3}$ corresponds to reasonably high molecular diffusion. $T = 10^{-4}$ corresponds to highly-oscillating flow field sampled at a frequency of 10 kHz.

The data generated based on the above parameters represents preferential and incomplete reactive-mixing. The system always starts in the unmixed state. But depending on the choice of the model parameters, it can result in either preferential mixing or incomplete mixing states. Incomplete mixing corresponds to a parameter scenario where $\frac{\alpha_L}{\alpha_T} = 10^4$ with large-scale vortex structures (e.g., $k_f L \leq 3$; see Fig. 4(a), Fig. 6(a)). This results in regions/islands (e.g., specifically near the center of the vortex) where product C concentration is zero. Preferential mixing represents a situation when $\frac{\alpha_L}{\alpha_T} = 10^4$ with small-scale vortex structures (e.g., $k_f L \geq 4$, Fig. 8(a), Fig. 10(a)). In this case, the system is not uniformly mixed due to anisotropy and small-scale features in flow field facilitate mixing along a certain direction. As a result, training and testing our CNN-LSTM models on this data allows us to see how well they predict different spatial-temporal patterns in mixing under high-anisotropy at late times. The reason to choose large anisotropic contrast data is because of its rich information content on the spatial-temporal evolution of reactive-mixing patterns (e.g., incomplete mixing, preferential mixing, interfacial mixing). Such patterns are not formed under lower anisotropic contrast. This is because the system tends to move towards uniform or well-mixed state (e.g., when $\frac{\alpha_L}{\alpha_T} \leq 100$) even in the early times [7,34].

4.3 Accuracy of mixing patterns under the proposed DL-based framework

In this subsection, we present detailed analysis and associated accuracy of the trained models. Prediction accuracy and its metrics are focused on how well the proposed CNN-LSTM models forecast mixing patterns under the combined effects of high anisotropy and different characteristic scales of vortex structures.

Fig. 4 compares the ground truth and deep learning model predictions at $t = 1.0$. The ground truth is based on reactive-transport simulation data generated for $\kappa_f L = 2$. This value of $\kappa_f L$ corresponds to reactive-mixing under large-scale vortex structures. The pro-



(a) Ground truth at $t=1.0$

(b) Prediction: 8% data (c) Prediction: 16% data (d) Prediction: 24% data (e) Prediction: 32% data

(f) Prediction: 40% data (g) Prediction: 48% data (h) Prediction: 56% data (i) Prediction: 64% data

(j) Prediction: 72% data (k) Prediction: 80% data

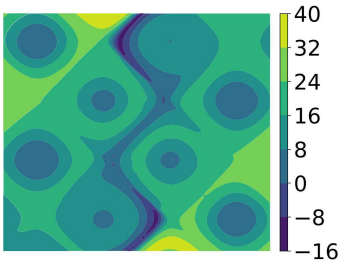
Figure 4: **Predictions for $\kappa_f L=2$:** This figure shows the ground truth from the non-negative FEM and predictions from the trained non-negative CNN-LSTM models at $t=1.0$. From this figure, it is evident that our deep learning model is able to effectively capture product C evolution with minimal training data (e.g., greater than or equal to 32%).

posed scenario also corresponds to incomplete mixing that occurs due to the combined effects of high anisotropy contrast along with large-scale features in velocity field. Specifically, we can find a total of eight regions/islands in the entire domain (e.g., vortex cen-

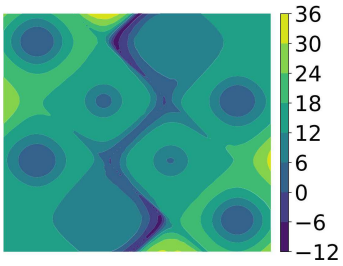
ters), where progress of mixing and formation of product C is very slow. At the center of these islands, the value of $c_C(x,y)$ is equal to zero. We analyze whether our trained CNN-LSTM models can capture such incomplete mixing patterns that occur due to large-scale features in the velocity field.

As mentioned in the prior section, FEM simulation data were obtained from $t=0.0$ to $t=1.0$ for a time increment of 0.001. Fig. 4(a) shows the ground truth at the end of simulation time (at $t=1.0$). Figs. 4(b)-(k) show the predictions at $t=1.0$ from the CNN-LSTM models trained using incrementally larger amount of training data as described in the individual captions. To elaborate, the forecast at $t=1.0$ in Fig. 4(b) is obtained using the CNN-LSTM model trained using the first 8% (from $t=0.001$ to $t=0.08$) of the ground truth data. The trained model then predicts $c_C(x,y)$ for the rest of the simulation time (i.e., $t=0.09, 0.10, \cdots, 1.00$). The CNN-LSTM model makes this forecast based on the procedure summarized in Fig. 2. It can be observed in Fig. 4 that the forecasts are generally better for higher amounts of training data. A very accurate prediction is obtained when we use 88% of the ground truth data. For this scenario, based on Fig. 5(k), the maximum possible prediction error is less than 6% in the entire domain. Moreover, the maximum possible prediction error to accurately capture interfacial mixing is less than 2%. However, qualitatively, we can see that non-negative CNN-LSTM model trained using ground truth data as low as 32% is able to effectively capture the underlying mixing pattern (e.g., interfacial mixing).

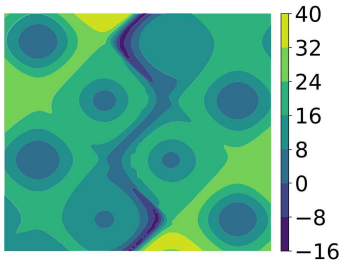
Fig. 5 compares the prediction errors in the entire domain at $t=1.0$. The prediction error is in percentage, which is calculated by taking the difference between the ground truth and the non-negative CNN-LSTM model prediction at every point in the domain. The error was calculated as *true value* minus *predicted value* and therefore a positive value of the error shows that the CNN-LSTM model under-predicts the concentration while negative error shows over prediction. From Fig. 5, it is clear that there is a coherent spatial structure in the error. It also shows where CNN-LSTM model fails to make an accurate prediction. That is, qualitatively and quantitatively, the error sheds light on the difficulties faced by the CNN-LSTM model on predicting mixing patterns with different amounts of training data. For instance, with 8% training data, we can see that the maximum possible value for under predictions and over predictions is approximately 40% and 16%, respectively. In this scenario, high-values of over prediction ($>30\%$) by the CNN-LSTM model are seen in the regions closer to the boundary of the domain (e.g., due to boundary effects). A large error at the boundary can be caused by the combined effects of convolution and pooling. This error can be reduced by designing tailored convolutional kernels (e.g., boundary filters) that can handle the boundary effects [28, 29]. Similarly, high-values of under prediction ($>10\%$) are seen near the mixing interface (e.g., due to local-scale features). This is where the barrier is removed and species are allowed to interact and facilitate the progress of chemical reaction. From Figs. 5(a)-(j), it is evident that to capture the interfacial mixing (e.g., due to high anisotropy) within an error of 10% (e.g., measured using the infinity norm), 32% of ground truth data is sufficient. To capture global-scale mixing features (e.g., resulting from molecular diffusion)



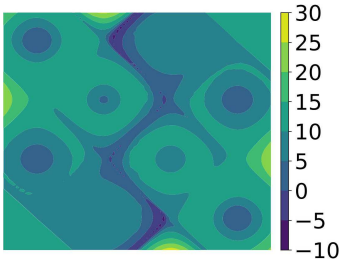
(a) Prediction error: 8% data



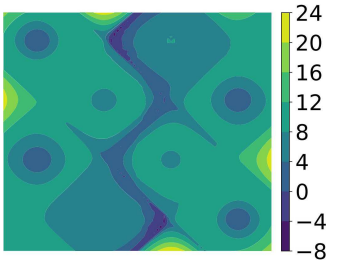
(b) Prediction error: 16% data



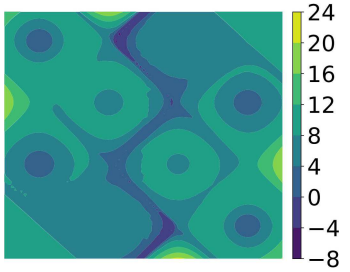
(c) Prediction error: 24% data



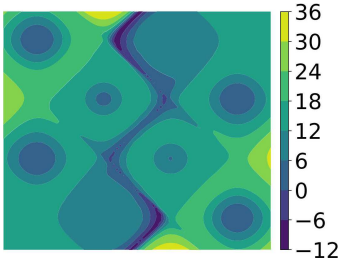
(d) Prediction error: 32% data



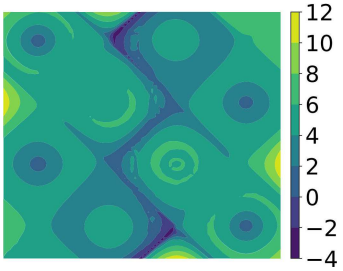
(e) Prediction error: 40% data



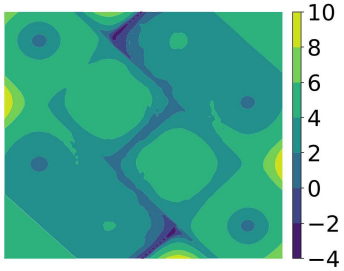
(f) Prediction error: 48% data



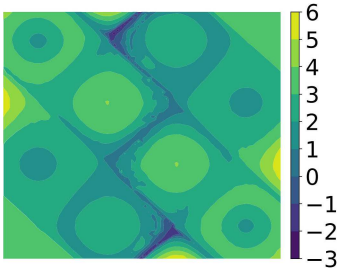
(g) Prediction error: 56% data



(h) Prediction error: 64% data



(i) Prediction error: 72% data



(j) Prediction error: 80% data

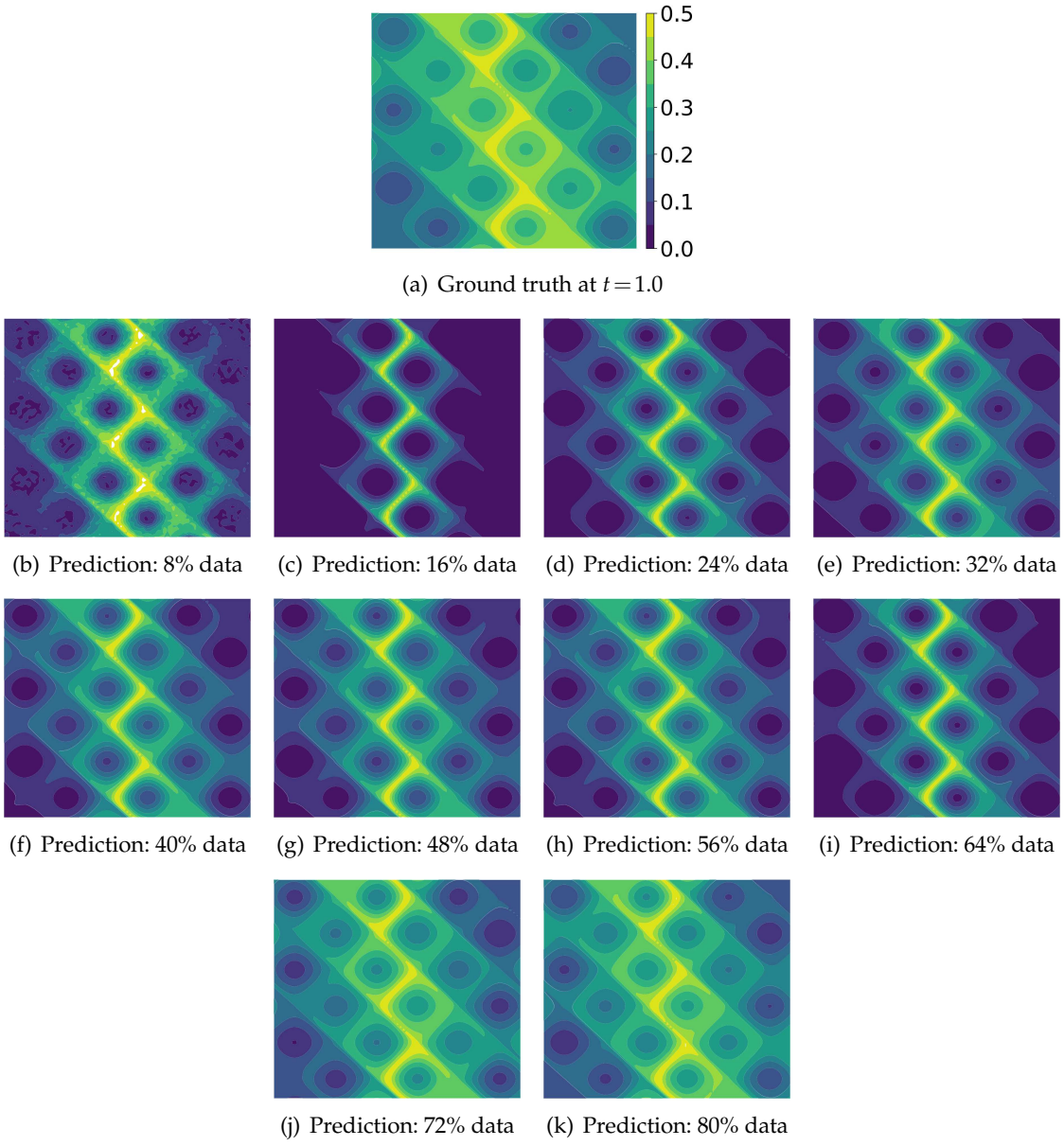
Figure 5: **Prediction error in percentage for $\kappa_f L=2$:** This figure compares the prediction errors in the entire domain at $t=1.0$ for different amount of training data. Based on the error values (e.g., $\leq 10\%$), it is evident that with less than 64% of training data, we can accurately capture different mixing patterns (e.g., interfacial mixing, mixing near vortices due to molecular diffusion).

within an error of 10%, 64% of ground truth data is needed. With this amount of training data, the prediction error associated with capturing local-scale features near the interface is reduced by more than 50%. Specifically, with 64% ground truth data, the prediction error to capture the interfacial mixing is less than 4%. This level of enhanced accuracy (e.g., using 32%, 64%, 88% data) instills confidence in the proposed CNN-LSTM model architecture to make reasonably good forecasts in both early and late times of reactive-mixing. With minimal data (between 32% to 64%), we can capture various patterns and associated multi-scale features in incomplete mixing arising due to large-scale vortex structures.

Fig. 6 shows the ground truth and the trained CNN-LSTM model predictions for $\kappa_f L = 3$. Similar to $\kappa_f L = 2$ scenario, we have large-scale vortex structures (≈ 18 vortices in the domain). However, the spatial-scale of these vortices for $\kappa_f L = 3$ are smaller compared to $\kappa_f L = 2$. As a result, the extent of incomplete mixing is reduced when compared to Fig. 4(a). This is because of the reduced size in the vortices, which improves the interaction between reactants. As a result, we have a better mixing scenario compared to $\kappa_f L = 2$ even under high anisotropy. Fig. 6(a) has six regions/islands where progress of mixing and formation of product C is very slow. These regions are located primarily near the boundary and at the corners of the domain (e.g., $(x,y) \approx (0.1,0.25), (0.1,0.6), (0.25,0.1), (0.75,0.9), (0.9,0.75), (0.9,0.4)$). Compared to Fig. 4(a), we have a lesser number of regions/islands where $c_C(x,y) = 0.0$. Additionally, the interfacial mixing in Fig. 6(a) is stronger than Fig. 4(a). This is because of larger number of vortices in the domain, which produce enhanced mixing.

Fig. 7 compares the prediction errors at the end of simulation time. Similar to $\kappa_f L = 2$, it is evident that the model forecasts are better for higher amounts of training data. The best and very accurate forecast is obtained when we use 80% of the ground truth data (see Fig. 6(k)). For 80% training data, the maximum possible prediction error is less than 8% in the entire domain. This accuracy in prediction is greater than that of $\kappa_f L = 2$ scenario (e.g., see Fig. 5(j)). The maximum possible prediction error to accurately capture interfacial mixing is less than 3% (double that of $\kappa_f L = 2$). In a nutshell, qualitatively and quantitatively, we can see that non-negative CNN-LSTM model trained using ground truth data, which is greater than or equal to 72% is able to effectively capture various patterns for enhanced mixing. This includes both interfacial mixing and mixing near the vortices.

Fig. 8 shows the mixing patterns under small-scale vortex structures (≈ 32 vortices in the domain). In this scenario, we observe preferential mixing aligned approximately at an angle of 15^o anti-clockwise to vertical barrier. This preferential direction consists of four small-scale vortices, which allow the reactant to mix faster across the interface when compared to $\kappa_f L = 2$ and 3. As a result, concentration of product C is non-zero in the entire domain. However, we can see the formation of the product is not homogeneous in the entire domain despite the small-scale features in the velocity field. This spatial inhomogeneity of the product formation is because of high contrast in anisotropy, which results in preferential mixing patterns. Qualitatively and quantitatively, the predictions shown in Figs. 8-9 shed light on how the CNN-LSTM model is able to capture different



(a) Ground truth at $t=1.0$

(b) Prediction: 8% data (c) Prediction: 16% data (d) Prediction: 24% data (e) Prediction: 32% data

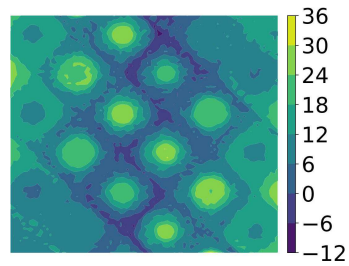
(f) Prediction: 40% data (g) Prediction: 48% data (h) Prediction: 56% data (i) Prediction: 64% data

(j) Prediction: 72% data (k) Prediction: 80% data

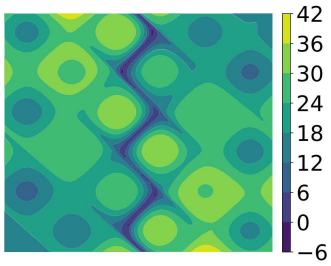
Figure 6: Predictions for $\kappa_f L=3$: This figure compares the ground truth and predictions from the trained non-negative CNN-LSTM models at $t=1.0$. From this figure, it is evident that 80% data is necessary to accurately capture multi-scale mixing patterns. This includes both interfacial mixing and mixing near the vortices.

mixing patterns. For example, 72–80% data is reasonably sufficient to capture product formation with 10% error in the entire domain. For this level of input data for training and prediction, we can accurately capture the interfacial mixing within 3–4%.

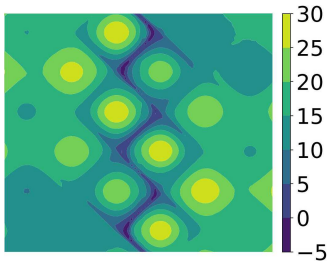
Fig. 10 compares the ground truth and predictions for $\kappa_f L=5$. This value of $\kappa_f L$



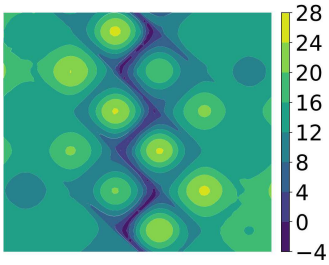
(a) Prediction error: 8% data



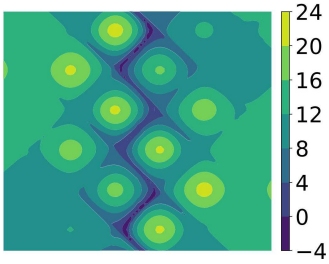
(b) Prediction error: 16% data



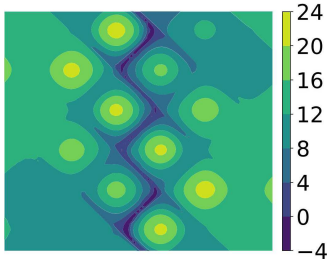
(c) Prediction error: 24% data



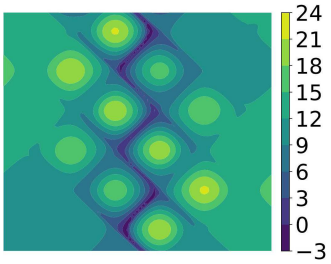
(d) Prediction error: 32% data



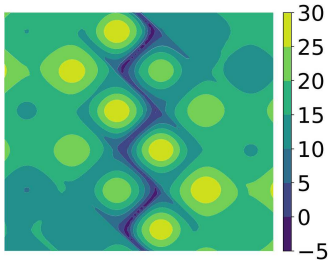
(e) Prediction error: 40% data



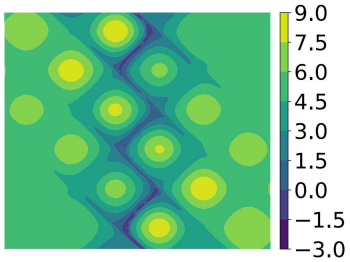
(f) Prediction error: 48% data



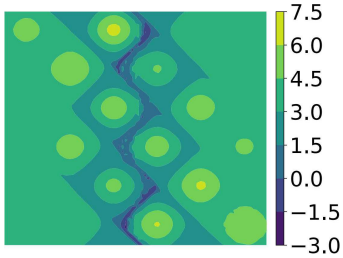
(g) Prediction error: 56% data



(h) Prediction error: 64% data

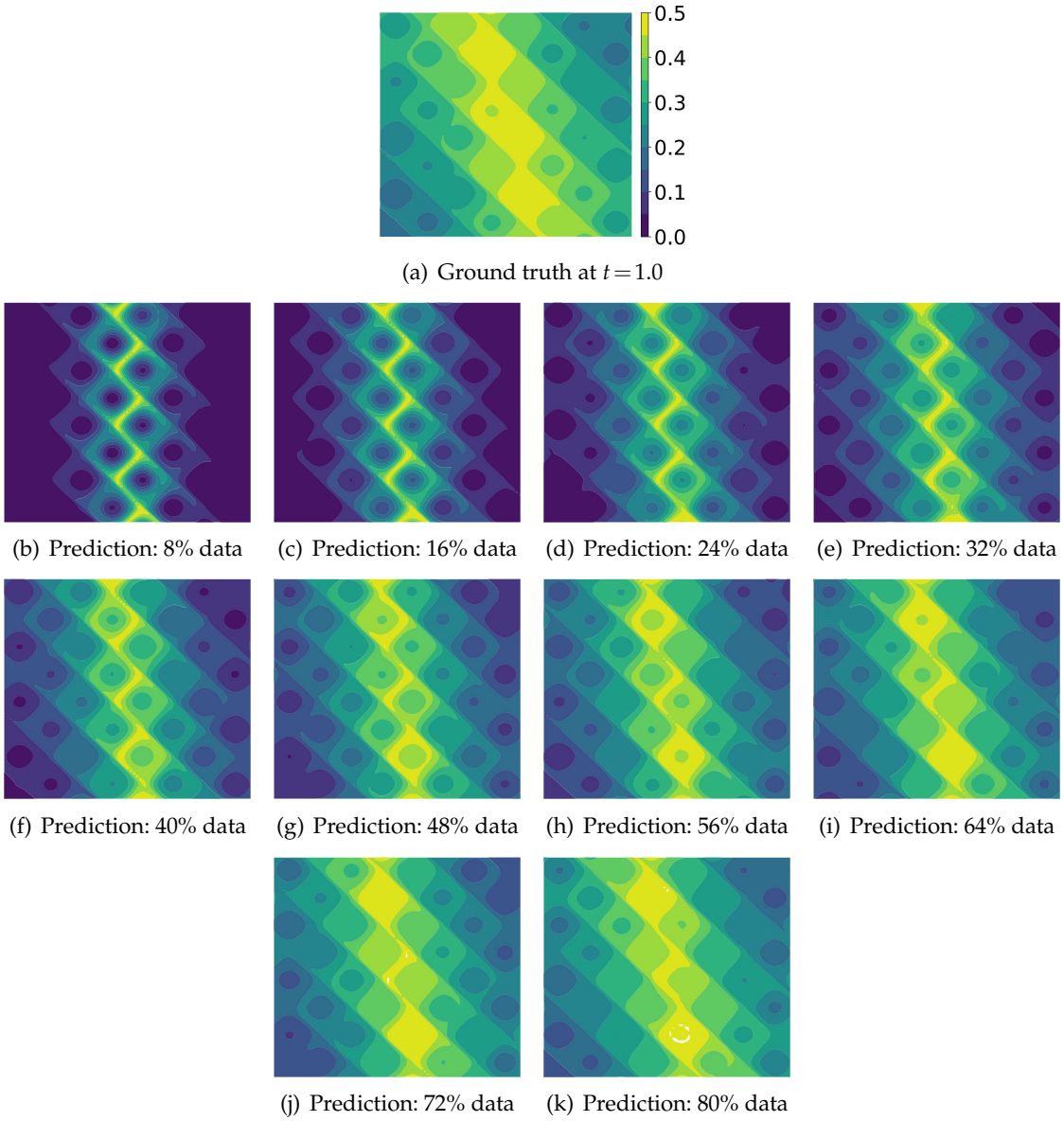


(i) Prediction error: 72% data



(j) Prediction error: 80% data

Figure 7: Prediction error in percentage for $\kappa_f L = 3$: This figure compares the prediction errors in the entire domain at $t = 1.0$ for different amount of training data. Based on the error values (e.g., $\leq 10\%$), it is evident that with less than 80% of training data is needed to accurately forecast product C formation.



(a) Ground truth at $t=1.0$

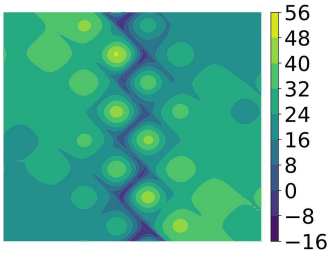
(b) Prediction: 8% data (c) Prediction: 16% data (d) Prediction: 24% data (e) Prediction: 32% data

(f) Prediction: 40% data (g) Prediction: 48% data (h) Prediction: 56% data (i) Prediction: 64% data

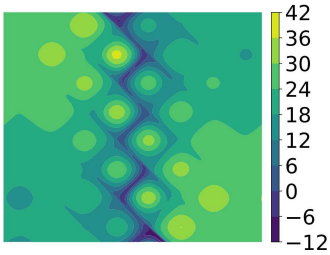
(j) Prediction: 72% data (k) Prediction: 80% data

Figure 8: Predictions for $\kappa_f L=4$: This figure compares the ground truth and predictions from the trained non-negative CNN-LSTM models at $t=1.0$. From this figure, it is evident that 72% ground truth data is reasonably sufficient to qualitatively capture different mixing patterns (e.g., preferential mixing due to small-scale features in the velocity field).

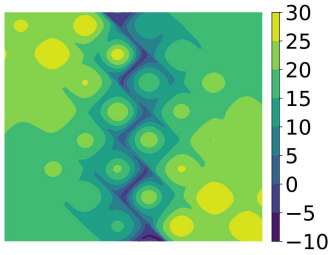
corresponds to preferential mixing under fine-scale features in the velocity field (≈ 50 vortices in the domain). The angle of preferential mixing is approximately 20–23^o anti-clockwise to vertical barrier. In this direction, the maximum possible plume width (e.g., the region in the domain that has $c_C(x,y) \geq 0.45$) is approximately 0.2–0.25 times the



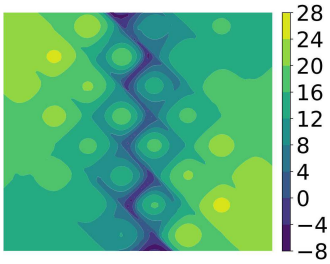
(a) Prediction error: 8% data



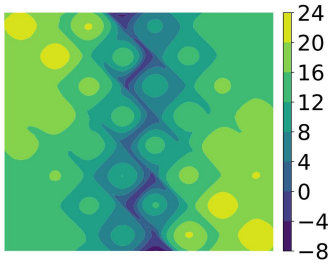
(b) Prediction error: 16% data



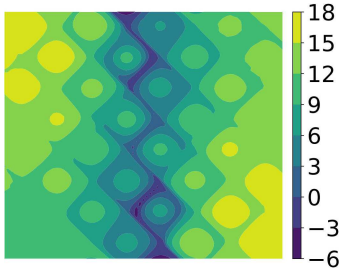
(c) Prediction error: 24% data



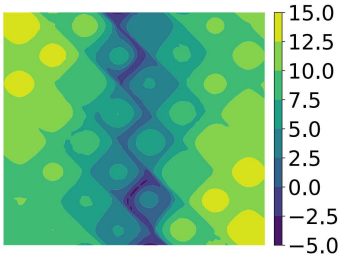
(d) Prediction error: 32% data



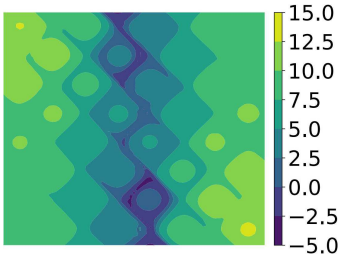
(e) Prediction error: 40% data



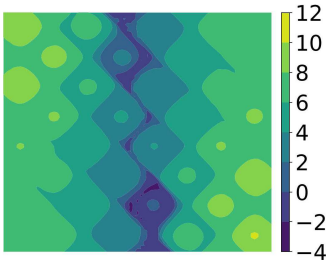
(f) Prediction error: 48% data



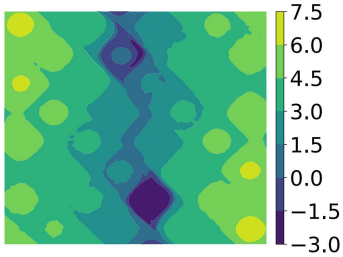
(g) Prediction error: 56% data



(h) Prediction error: 64% data

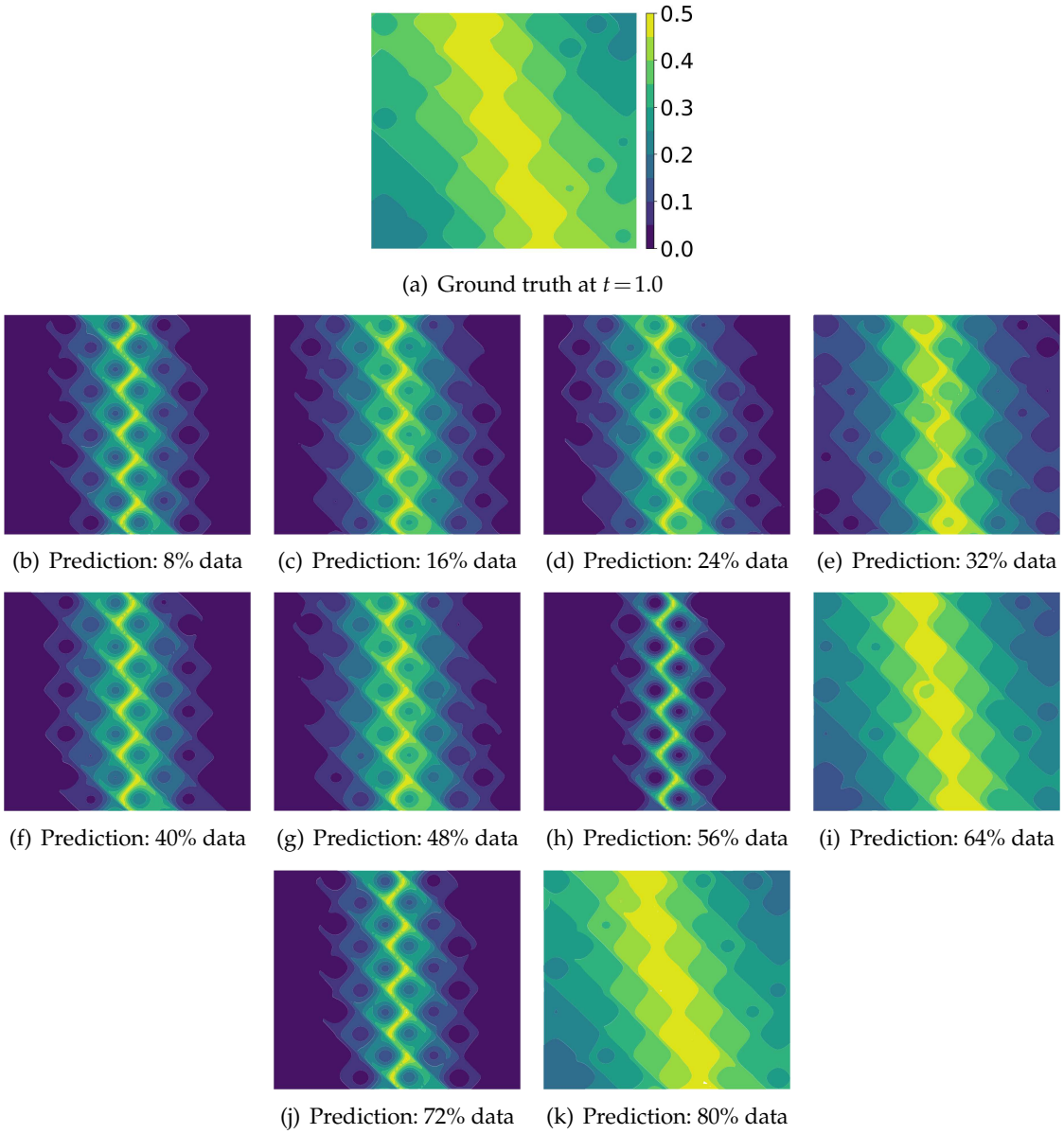


(i) Prediction error: 72% data



(j) Prediction error: 80% data

Figure 9: Prediction error in percentage for $\kappa_f L = 4$: This figure compares the prediction errors in the entire domain at $t = 1.0$ for different amount of training data. Based on the error values (e.g., $\leq 10\%$), it is evident that with 80% training data, we can accurately capture small-scale mixing patterns with an accuracy less than 10%.



(a) Ground truth at $t=1.0$

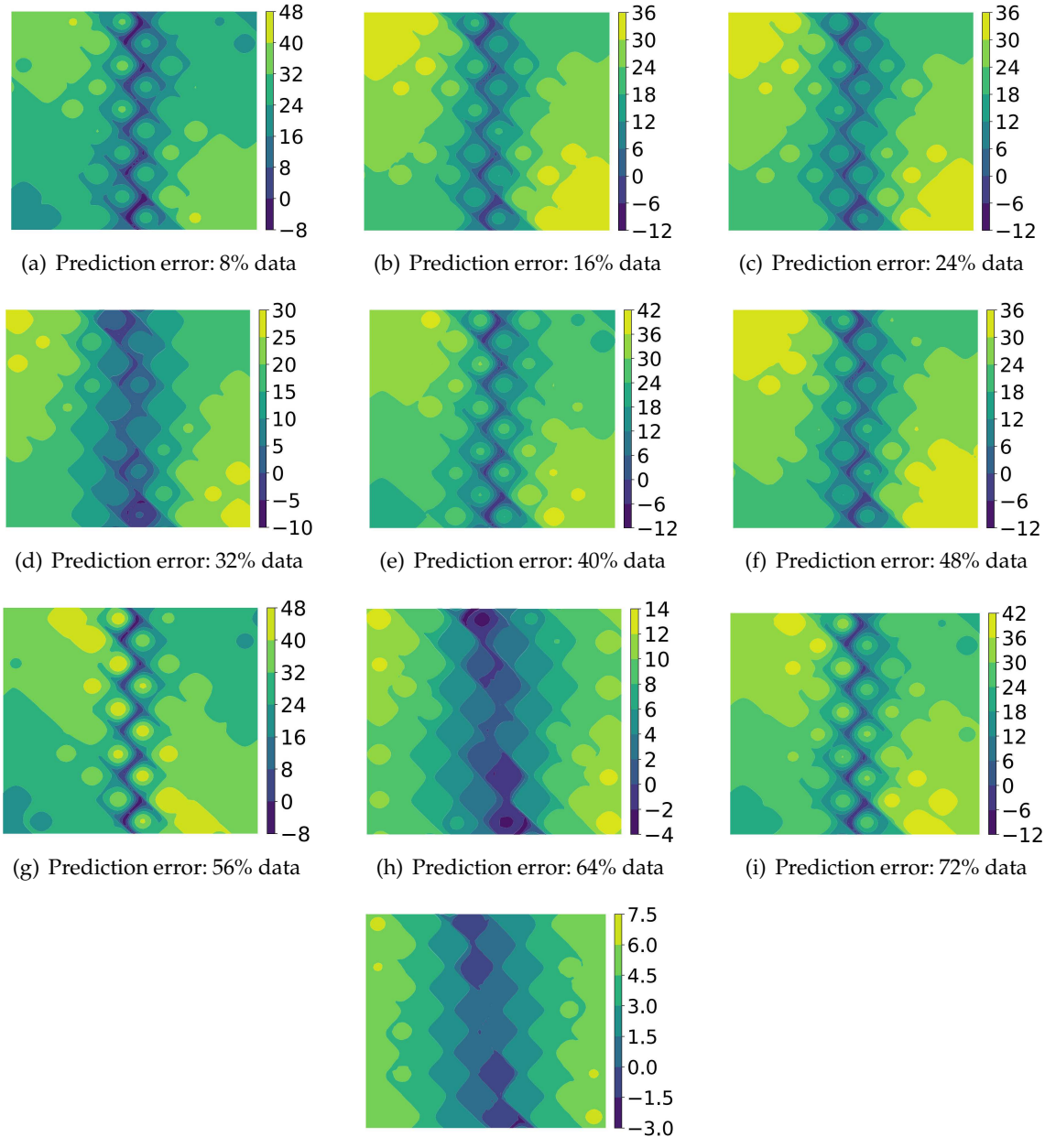
(b) Prediction: 8% data (c) Prediction: 16% data (d) Prediction: 24% data (e) Prediction: 32% data

(f) Prediction: 40% data (g) Prediction: 48% data (h) Prediction: 56% data (i) Prediction: 64% data

(j) Prediction: 72% data (k) Prediction: 80% data

Figure 10: Predictions for $\kappa_f L=5$: This figure compares the ground truth and predictions from the trained non-negative CNN-LSTM models at $t=1.0$. From this figure, it is evident that 64% ground truth data is needed to reasonably capture product formation in the entire domain (e.g., preferential mixing patterns, mixing away from interface).

length of the domain. Based on Figs. 10(i)-(k), the CNN-LSTM model prediction based on 64–80% ground truth data is able to capture most of the characteristics of this plume (e.g., plume width, angle of preferential mixing). Figs. 11(h)-(j) show that the associated



(a) Prediction error: 8% data (b) Prediction error: 16% data (c) Prediction error: 24% data

(d) Prediction error: 32% data (e) Prediction error: 40% data (f) Prediction error: 48% data

(g) Prediction error: 56% data (h) Prediction error: 64% data (i) Prediction error: 72% data

(j) Prediction error: 80% data

Figure 11: Prediction error in percentage for $\kappa_f L = 5$: This figure compares the prediction errors in the entire domain at $t = 1.0$ for different amount of training data. Based on the error values (e.g., $\leq 10\%$), it is evident that with 80% training data, we can accurately capture product formation in the entire domain. To accurately predict preferential mixing patterns (e.g., with an accuracy less than 5%), 64% of the ground truth is sufficient.

prediction errors (e.g., measured in infinity norm) are less than 5%. This accuracy instill confidence in the predictive capability of our models to forecast mixing patterns at late times under the combined effect of high anisotropy and fine-scale features in the velocity field.

4.4 Computational cost

The computational cost to run a single high-fidelity numerical simulation is approximately 26 mins (1560 seconds) on a single core and 18 mins (1080 seconds) on a eight core processor (Intel(R)Xeon(R) CPU E5-2695 v4 2.10GHz) [34]. The simulation dataset was developed using high-performance computing (HPC) resources [7, 34, 52]. However, access to leadership class supercomputing resources (e.g., NERSC [2], ALCF [1], OLCF [3]) to researchers is generally limited. Additionally, even after having access to such HPC systems, there are various challenges and bottlenecks to generate data for longer-periods of simulation times. This includes availability of compute time to users (e.g., nodes, machine hours), waiting time in job submission queue, and cap on the maximum amount of wall clock time to run on a supercomputer. For example, at NERSC user facility, one has to restart simulations every two days as the cap to run on these machines is 48 hours in a regular queue. Due to this bottleneck, simulations need to be restarted a lot of times to generate data at future time-steps. As a result, the overall wall clock time to generate simulation data for multiple realizations can be in the order of weeks to months. Our DL-based framework provides an attractive way to overcome these challenges. Specifically, this computational cost can be avoided by training the proposed DL models on partially generated data to forecast spatial-temporal variables at future times.

We have trained the proposed CNN-LSTM models on a MacBook Pro Laptop (2GHz 2-Core Intel i7 CPU, 8GB DDR4 RAM). The associated computational cost to train the CNN-LSTM models and make a prediction is shown in Table 1. From this table, it is evident that model training is computationally expensive as the wall clock times are in the $\mathcal{O}(10^3)$ to $\mathcal{O}(10^4)$ seconds. This is expected as CPUs alone (with low-end memory) are really slow for training deep learning models. However, once the CNN-LSTM model is developed, the wall clock time to make an inference (i.e., predicting the spatial concentration of product C at the next time-step) is very low. That is, the inference time on this laptop is in the order of $\mathcal{O}(10^{-1})$ seconds. This shows that our proposed CNN-LSTM model is fast in addition to its predictive capability. Our future work involves accelerating the training process using Google Cloud Platform's GPUs and TPUs [12].

We note that it is straightforward to perform similar analysis for other realizations (e.g., for different parameters in reaction-diffusion model). This can be achieved by re-training the proposed non-negative CNN-LSTM models with minimal data on other input parameters. The re-training process can be minimal as the weights of the convolutional layers that create the feature maps are not updated. This is because these CNN layers learned the underlying representation of the reactive-transport process through the proposed training approach. Specifically, minimal re-training can be performed through

Table 1: Training wall clock time (TWCT) and prediction wall clock time (PWCT) for the proposed non-negative CNN-LSTM models. The reported TWCT and PWCT are in seconds. TWCT is the amount of time taken to ingest the simulation data and perform the training process. PWCT is the average amount of time taken to make a spatial prediction of the concentration field per time-step for all the future times. As mentioned in Section 3.2, CNN-LSTM model are trained using 100 epochs with a batch size of 6.

Training data	**TWCT**				**Avg. PWCT (all $\kappa_f L$)**
	$\kappa_f L=2$	$\kappa_f L=3$	$\kappa_f L=4$	$\kappa_f L=5$	
8%	3239	3239	3138	3037	0.1325
16%	6339	6037	6037	6037	0.1326
24%	8937	9037	9238	8937	0.1327
32%	11937	11837	11836	11937	0.1330
40%	14837	14837	14837	15137	0.1333
48%	17837	17736	17736	17837	0.1334
56%	20736	20636	21037	20636	0.1335
64%	23626	23536	23636	23737	0.1343
72%	26737	26436	26636	25835	0.1346
80%	29737	29737	29837	29436	0.1365
88%	32536	32436	32436	32436	0.1395
96%	35436	35336	35236	35536	0.1400

transfer learning techniques, where weights of the dense layer connected to the output are only updated. One should expect similar outcomes as described in this study.

In this study, we note that the simulation time for running a single realization is lower than the training time due to a small number of degrees of freedom. As the focus of the current study is to show the feasibility of our proposed approach, we have chosen a proof-of-concept problem and a simpler mesh size. However, as the mesh size increases, the computational cost to run a single realization increases exponentially (if the computing resources are limited), which might take days to weeks of wall clock time. Developing the non-negative CNN-LSTM model takes a similar time because the architecture details remain the same, and data is segmented to accelerate the training process. Hence, the computational cost of training the model and obtaining predictions from the trained model will be of the same order as mentioned here after accounting for the different computer processor speed.

Note that the prediction accuracy can further be improved by at least the following two ways: (1) Increase in training data: Given sufficiently long-periods of data, a larger set of training data will further improve the accuracy. For instance, we can use this data to better tune the proposed CNN-LSTM model and improve the accuracy. (2) Changing the neural network architecture: Another way to achieve improved accuracy is to use recent advances in neural network architectures (e.g., that utilize attention mechanisms [51]). Examples include transformers [18,54]. Attention-based neural networks [20] (e.g.,

transformers) can help us improve the accuracy and better capture the mixing patterns at the interfaces.

Finally, we emphasize that the current framework is extensible in higher dimensions, e.g., 3D. This can be achieved by using 3D convolution, which is readily available in deep learning packages such as Tensorflow [5] and PyTorch [4]. As already stated in this paper, if data from numerical methods are unavailable, data from other sources like experiments, distributed or point sensors, cameras, satellite images can be utilized for the proposed framework. If data from numerical methods and other aforementioned sources are unavailable. The recent advances in physics-informed neural networks (e.g., [44]) provide a viable route to develop a framework to solve mixing problems. This is currently an active research topic being pursued by the authors.

Below we summarize our results:

- Our model's prediction accuracy generally increases with the amount of training data.
- The maximum error locations are near the regions of imposed boundary condition and low concentration zones.
- The concentration profile in the interfacial mixing zones is captured with reasonably good accuracy by our non-negative CNN-LSTM model trained using less than 32% of the simulation data.
- The high concentration zones are accurately captured within 10% error (measured in infinity norm) by training our model using $\approx 64\%$ of the simulation data.
- The wall clock time to make prediction/inference is $\approx \mathcal{O}(10^{-5})$ to that of running a high-fidelity simulation.

4.5 A potential application

Based on the accuracy of the results presented in Sections 4.3-4.4, the proposed DL framework will be an ideal candidate for modeling virus transport. Several prior studies have used reactive-transport for pandemic modeling. Some concrete example include: predict the evolution of COVID-19 [30,53], and capture the transport of microbial pathogens in the terrestrial subsurface (primarily at aquatic interfaces) [13, 23, 24]. In the said applications, accurate forecasts of reactive-transport are essential because fluctuating flow regimes (e.g., water table fluctuations, flow fields given by Eqs. (4.3)-(4.4)) can enhance virus detachment from solid-water interfaces and air-water interfaces [57]. These flow fields can result in regions (e.g., interfacial mixing areas as shown in Figs. 4, 6, 8, 10) that have high concentrations in drinking water resources, which the cited references did not consider. Our DL framework can be applied to solve such problems.

The non-negative CNN-LSTM models would be able to accurately model the spatial-temporal evolution of an epidemic such as COVID-19 and virus transport in porous media. The DL-based model can be trained using the initial few days of epidemic evolution

over a region of interest. The trained model can then be used to predict the future evolution of the epidemic. This would provide very significant information to the policymakers in terms of when and where the epidemic infections would be reaching the critical level. This information would also help in planning the allocation of resources and in taking early actions to suppress the spread of the disease (e.g., lockdown, travel ban).

5 Concluding remarks

There is a need for fast, reliable, and physically meaningful forecasts of the concentrations of chemical species and the evolution of reactive-mixing for diverse engineering applications such as combustion and pandemic modeling. However, high-resolution numerical simulations mostly use the initial and boundary values and cannot utilize any intermediate temporal data to improve accuracy of the predictions. Secondly, high-resolution numerical simulations are often expensive, as each simulation may take several hours or days to provide spatial-temporal patterns of mixing, especially for problems with large spatial and temporal domains. To address the said need, we have presented a fast and predictive framework that can forecast the spatial-temporal evolution of reactive-transport using minimal data. Our approach leverages recent advances in deep learning techniques—non-negative constrained CNNs for capturing spatial spreading and LSTMs for temporal evolution. The framework was used to develop a new non-negative model, which elegantly overcomes the issue of spurious negative concentration predictions by traditional numerical methods like FEM.

The DL-based model developed herein needs to be trained using only a subset of the data for the total temporal domain of interest. The data can be obtained using high-fidelity simulations or experiments. Our results demonstrate that less than 32% of the simulation data is needed to capture the interfacial mixing (e.g., local-scale mixing features due to combined effects of anisotropy and vortex structures) within an error of 10% (as measured using the infinity norm). To capture global features at the same level of accuracy, less than 70–80% data is needed. The time needed for a forecast is a fraction ($\approx \mathcal{O}(10^{-6})$) of the time needed for training. Also, the ratio of the solution time for the proposed deep learning framework to that for the high-fidelity simulation used to generate data for training the model is $\approx \mathcal{O}(10^{-5})$.

Based on these results, we contend that the subject DL framework would be best suited to three types of cases: (1) Data obtained through expensive and/or time consuming experiments. (2) Data obtained through simulation for problems in which interfacial mixing results are more important than the results in the entire domain so that less than 32% of the training data would produce sufficient accuracy. (3) Data obtained through simulation for problems in which global accuracy is important *and* the simulation solution time per time step is so large that obtaining $\approx$70% of the data for training purpose would still save significant amount of solution time. It is well-known that problems with extremely large solution times are not uncommon in reactive-transport domain.

The efficiency of the developed framework can further be improved by using graphical processing units (GPUs) and tensor processing units (TPUs) [12]. Distributed training (e.g., using Horovod [48]) and scalable machine learning (e.g., using DeepHyper [10]) also provide viable routes to achieve speedup by splitting the workload to train the non-negative CNN-LSTM model among multiple processors (i.e., worker nodes). Thus, the accuracy of the predictions combined with the fast inference makes our non-negative CNN-LSTM models attractive for real-time forecasting of species concentration in reactive transport problems for a wide range of engineering applications.

Datasets

The datasets used in this study were developed as a part previously published study [7, 34, 52]; these works are cited and acknowledged in this paper.

Acknowledgments

MKM is supported by ExaSheds project during the paper writing process, which was supported by the U.S. Department of Energy, Office of Science, under Award Number DE-AC02-05CH11231. MKM also thanks Dr. Satish Karra for his help on data generation. KBN acknowledges the support from the University of Houston through High Priority Area Research Seed Grant.

Appendix A: Generation of non-negative solutions

For completeness, this section provides details on the non-negative single-field formulation used to generate data for training and testing the CNN-LSTM models.

Let the total time interval $[0,\mathcal{I}]$ be discretized into N non-overlapping sub-intervals:

$$[0,\mathcal{I}] = \bigcup_{n=1}^{N} [t_{n-1}, t_n], \tag{A.1}$$

where $t_0 = 0$ and $t_N = \mathcal{I}$. Assuming a uniform time step Δt, we define:

$$c_F^{(n)}(\mathbf{x}) := c_F(\mathbf{x}, t_n) \quad \text{and} \quad c_G^{(n)}(\mathbf{x}) := c_G(\mathbf{x}, t_n). \tag{A.2}$$

Following [40], we will first discretize with respect to time the governing equations for the invariants; we will use the backward Euler time-stepping scheme. The resulting time-

discrete equations corresponding to Eqs. (2.6a)-(2.6d) are:

$$\left(\frac{1}{\Delta t}\right) c_F^{(n+1)}(\mathbf{x}) - \operatorname{div}\left[\mathbf{D}(\mathbf{x}) \operatorname{grad}\left[c_F^{(n+1)}(\mathbf{x})\right]\right] = f_F(\mathbf{x},t_{n+1}) + \left(\frac{1}{\Delta t}\right) c_F^{(n)}(\mathbf{x}) \qquad \text{in } \Omega, \tag{A.3a}$$

$$c_F^{(n+1)}(\mathbf{x}) = c_F^{\mathrm{p}}(\mathbf{x},t_{n+1}) := c_A^{\mathrm{p}}(\mathbf{x},t_{n+1}) + \left(\frac{n_A}{n_C}\right) c_C^{\mathrm{p}}(\mathbf{x},t_{n+1}) \qquad \text{on } \Gamma^{\mathrm{D}}, \tag{A.3b}$$

$$\mathbf{n}(\mathbf{x}) \bullet \mathbf{D}(\mathbf{x}) \operatorname{grad}\left[c_F^{(n+1)}(\mathbf{x})\right] = h_F^{\mathrm{p}}(\mathbf{x},t_{n+1}) := h_A^{\mathrm{p}}(\mathbf{x},t_{n+1}) + \left(\frac{n_A}{n_C}\right) h_C^{\mathrm{p}}(\mathbf{x},t_{n+1}) \quad \text{on } \Gamma^{\mathrm{N}}, \tag{A.3c}$$

$$c_F(\mathbf{x},t_0) = c_F^0(\mathbf{x}) := c_A^0(\mathbf{x}) + \left(\frac{n_A}{n_C}\right) c_C^0(\mathbf{x}) \qquad \text{in } \overline{\Omega}. \tag{A.3d}$$

Note that one can extend the proposed framework to other time-stepping schemes by following the non-negative procedures given in [40]. The standard finite element Galerkin formulation for Eqs. (A.3a)-(A.3d) is given as follows: Find $c_F^{(n+1)}(\mathbf{x}) \in \mathcal{C}_F^t$ with

$$\mathcal{C}_F^t := \left\{ c_F(\bullet,t) \in H^1(\Omega) \,\middle|\, c_F(\mathbf{x},t) = c_F^{\mathrm{p}}(\mathbf{x},t) \text{ on } \Gamma^{\mathrm{D}} \right\} \tag{A.4}$$

such that we have

$$\mathcal{B}^t\left(w; c_F^{(n+1)}\right) = L_F^t(w) \quad \forall w(\mathbf{x}) \in \mathcal{W}, \tag{A.5}$$

where the bilinear form and linear functional are, respectively, defined as

$$\mathcal{B}^t\left(w; c_F^{(n+1)}\right) := \frac{1}{\Delta t} \int_\Omega w(\mathbf{x})\, c_F^{(n+1)}(\mathbf{x})\, \mathrm{d}\Omega + \int_\Omega \operatorname{grad}[w(\mathbf{x})] \bullet \mathbf{D}(\mathbf{x}) \operatorname{grad}[c_F^{(n+1)}(\mathbf{x})]\, \mathrm{d}\Omega, \tag{A.6a}$$

$$L_F^t(w) := \frac{1}{\Delta t} \int_\Omega w(\mathbf{x})\, c_F^{(n)}(\mathbf{x})\, \mathrm{d}\Omega + \int_{\Gamma^{\mathrm{N}}} w(\mathbf{x})\, h_F^{\mathrm{p}}(\mathbf{x},t_{n+1})\, \mathrm{d}\Gamma, \tag{A.6b}$$

and

$$\mathcal{W} := \left\{ w(\mathbf{x}) \in H^1(\Omega) \,\middle|\, w(\mathbf{x}) = 0 \text{ on } \Gamma^{\mathrm{D}} \right\}. \tag{A.7}$$

A similar formulation can be written for the invariant c_G.

The Galerkin formulation given by Eq. (A.5) can be re-written as a minimization problem as follows:

$$\underset{c_F^{(n+1)}(\mathbf{x}) \in \mathcal{C}_F^t}{\text{minimize}} \quad \frac{1}{2}\mathcal{B}^t\left(c_F^{(n+1)}; c_F^{(n+1)}\right) - L_F^t\left(c_F^{(n+1)}\right). \tag{A.8}$$

However, it has been shown that the Galerkin formulation leads to negative concentrations for product C [39], which is unphysical. Instead, we will solve the minimization problem Eq. (A.8) with the constraint that c_F is non-negative; that is,

$$c_F \geq 0. \tag{A.9}$$

A similar minimization problem with non-negative constraint for c_G can be written as follows:

$$\underset{c_G^{(n+1)}(\mathbf{x}) \in \mathcal{C}_G^t}{\text{minimize}} \quad \frac{1}{2}\mathcal{B}^t\left(c_G^{(n+1)}; c_G^{(n+1)}\right) - L_G^t\left(c_G^{(n+1)}\right) \tag{A.10a}$$

$$\text{subject to} \quad c_G \geq 0. \tag{A.10b}$$

Upon discretizing Eqs. (A.8)-(A.10a) using low-order finite elements, one arrives at:

$$\underset{\mathbf{c}_F^{(n+1)} \in \mathbb{R}^{ndofs}}{\text{minimize}} \quad \frac{1}{2}\left\langle \mathbf{c}_F^{(n+1)}; \mathbf{K}\mathbf{c}_F^{(n+1)} \right\rangle - \frac{1}{\Delta t}\left\langle \mathbf{c}_F^{(n+1)}; \mathbf{c}_F^{(n)} \right\rangle \tag{A.11a}$$

$$\text{subject to} \quad \mathbf{0} \preceq \mathbf{c}_F^{(n+1)}, \tag{A.11b}$$

$$\underset{\mathbf{c}_G^{(n+1)} \in \mathbb{R}^{ndofs}}{\text{minimize}} \quad \frac{1}{2}\left\langle \mathbf{c}_G^{(n+1)}; \mathbf{K}\mathbf{c}_G^{(n+1)} \right\rangle - \frac{1}{\Delta t}\left\langle \mathbf{c}_G^{(n+1)}; \mathbf{c}_G^{(n)} \right\rangle \tag{A.11c}$$

$$\text{subject to} \quad \mathbf{0} \preceq \mathbf{c}_G^{(n+1)}, \tag{A.11d}$$

where $\langle \bullet; \bullet \rangle$ represents the standard inner-product on Euclidean spaces, and "$ndofs$" denotes the (nodal) degrees-of-freedom. $\mathbf{c}_F^{(n+1)}$ and $\mathbf{c}_G^{(n+1)}$ are the nodal concentration vectors for the invariant c_F and c_G at time level t_{n+1}, respectively. The coefficient matrix $\mathbf{K}$ is positive definite, and hence a unique global minimizer exists [37].

References

[1] ALCF – Argonne Leadership Computing Facility, 2021. URL `https://www.alcf.anl.gov/`. Accessed on: 2021-07-19.

[2] NERSC – National Energy Research Scientific Computing Center, 2021. URL `https://www.nersc.gov/`. Accessed on: 2021-07-19.

[3] OLCF – Oak Ridge Leadership Computing Facility, 2021. URL `https://www.olcf.ornl.gov/`. Accessed on: 2021-07-19.

[4] PyTorch – An open source machine learning framework that accelerates the path from research prototyping to production deployment, 2021. URL `https://pytorch.org/`. Accessed on: 2021-07-19.

[5] Tensorflow – An end-to-end open source machine learning platform, 2021. URL `https://www.tensorflow.org/`. Accessed on: 2021-07-19.

[6] A. Adrover, S. Cerbelli, and M. Giona. A spectral approach to reaction/diffusion kinetics in chaotic flows. *Computers & Chemical Engineering*, 26:125–139, 2002. doi: 10.1016/S0098-1354(01)00761-X.

[7] B. Ahmmed, M. K. Mudunuru, S. Karra, S. C. James, and V. V. Vesselinov. A comparative study of machine learning models for predicting the state of reactive mixing. *arXiv preprint: 2002.11511*, 2020.

[8] A. Amikiya and M. Banda. Modelling and simulation of reactive transport phenomena. *Journal of Computational Science*, 28:155 – 167, 2018. doi: 10.1016/j.jocs.2018.08.002.

[9] T. Arbogas, S. Bryant, C. Dawson, F. Saaf, C Wang, and M. Wheeler. Computational methods for multiphase flow and reactive transport problems arising in subsurface contaminant remediation. *Journal of Computational and Applied Mathematics*, 74(1):19 – 32, 1996. doi: 10.1016/0377-0427(96)00015-5.

[10] P. Balaprakash, M. Salim, T. Uram, V. Vishwanath, and S. Wild. Deephyper: Asynchronous hyperparameter search for deep neural networks. In *2018 IEEE 25th international conference on high performance computing (HiPC)*, pages 42–51. IEEE, 2018. doi: 10.1109/HiPC.2018.00014.

[11] R. Balestriero. A spline theory of deep learning. In *International Conference on Machine Learning*, pages 374–383, 2018.

[12] E. Bisong. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer, Berlin, 2019.

[13] S. Bradford, V. Morales, W. Zhang, R. Harvey, A. Packman, A. Mohanram, and C. Welty. Transport and fate of microbial pathogens in agricultural settings. *Critical Reviews in Environmental Science and Technology*, 43(8):775–893, 2013. doi: 10.1080/10643389.2012.710449.

[14] J. Chang, S. Karra, and K. B. Nakshatrala. Large-scale optimization-based non-negative computational framework for diffusion equations: Parallel implementation and performance studies. *Journal of Scientific Computing*, 70:243–271, 2017. doi: 10.1007/s10915-016-0250-5.

[15] X. Chen, J. Duan, and G. Karniadakis. Learning and meta-learning of stochastic advection-diffusionreaction systems from sparse measurements. *European Journal of Applied Mathematics*, page 124, 2020. doi: 10.1017/S0956792520000169.

[16] F. Chollet. *Deep Learning with Python*. Manning Publications Company, Shelter Island, NY, 2017.

[17] R. Codina. On stabilized finite element methods for linear systems of convection–diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 188(1-3):61–82, 2000. doi: 10.1016/S0045-7825(00)00177-8.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, and N. Houlsby J. Uszkoreit. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[19] J. Gao, H. Xing, Z. Tian, J. Pearce, M. Sedeka, S. Golding, and V. Rudolphc. Reactive transport in porous media for CO2 sequestration: Pore scale modeling using the lattice Boltzmann method. *Computers and Geosciences*, 98:9 – 20, 2017. doi: 10.1016/j.cageo.2016.09.008.

[20] N. Geneva and N. Zabaras. Transformers for modeling physical systems. *arXiv preprint arXiv:2010.03957*, 2020.

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] A. Hansen, J. Peterson, J. Ellis, G. Sednek, and B. Wilson. Terrestrial-aquatic linkages: Understanding the flow of energy and nutrients across ecosystem boundaries. *Department of*

Fish, Wildlife, and Conservation Biology Colorado State University Fort Collins, Colorado 80526.

[24] R. Harvey, H. Harms, and L. Landkamer. Transport of microorganisms in the terrestrial subsurface: In situ and laboratory methods. In *Manual of Environmental Microbiology, Third Edition*, pages 872–897. American Society of Microbiology, Bel Air, MD, 2007.

[25] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 2012.

[26] Z. Huangwei, Z. Majie, and H. Zhiwei. Large eddy simulation of turbulent supersonic hydrogen flames with openfoam. *Fuel*, 282:118812, 2020. doi: 10.1016/j.fuel.2020.118812.

[27] W. Hundsdorfer and J. G. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, volume 33. Springer Science & Business Media, Berlin, 2013.

[28] C. Innamorati, T. Ritschel, T. Weyrich, and N. J. Mitra. Learning on the edge: Investigating boundary filters in CNNs. *International Journal of Computer Vision*, pages 1–10, 2019. doi: 10.1007/s11263-019-01223-y.

[29] M. Islam, M. Kowal, S. Jia, K. Derpanis, and N. Bruce. Boundary Effects in CNNs: Feature or Bug? 2020. URL `https://openreview.net/forum?id=M4qXqdw3xC`.

[30] P. K. Jha, L. Cao, and J. T. Oden. Bayesian-based predictions of COVID-19 evolution in Texas using multispecies mixture-theoretic continuum models. *Computational Mechanics*, pages 1–14, 2020. doi: 10.1007/s00466-020-01889-z.

[31] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint: 1412.6980*, 2014.

[32] A. Leal, S. Kyas, D. Kulik, and M. Saar. Accelerating reactive transport modeling: on-demand machine learning algorithm for chemical equilibrium calculations. *Transport in Porous Media*, 133(2):161–204, 2020. doi: 10.1007/s11242-020-01412-1.

[33] A. Li, R. Chen, A. Farimani, and Y. Zhang. Reaction diffusion system prediction based on convolutional neural network. *Scientific Reports*, 10(1):1–9, 2020. doi: 10.1038/s41598-020-60853-2.

[34] M. K. Mudunuru and S. Karra. Physics-informed machine learning models for predicting the progress of reactive mixing. *Computer Methods in Applied Mechanics and Engineering*, 374: 113560, 2021. doi: 10.1016/j.cma.2020.113560.

[35] M. K. Mudunuru and K. B. Nakshatrala. On enforcing maximum principles and achieving element-wise species balance for advection-diffusion-reaction equations under the finite element method. *Journal of Computational Physics*, 305:448–493, 2016. doi: 10.1016/j.jcp.2015.09.057.

[36] M. K. Mudunuru and K. B. Nakshatrala. On mesh restrictions to satisfy comparison principles, maximum principles, and the non-negative constraint: Recent developments and new results. *Mechanics of Advanced Materials and Structures*, 24:556–590, 2017. doi: 10.1080/15502287.2016.1166160.

[37] H. Nagarajan and K. B. Nakshatrala. Enforcing the non-negativity constraint and maximum principles for diffusion with decay on general computational grids. *International Journal for Numerical Methods in Fluids*, 67(7):820–847, 2011. doi: 10.1002/fld.2389.

[38] V. Nair and G. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, pages 807–814, 2010. URL `https://icml.cc/Conferences/2010/papers/432.pdf`.

[39] K. B. Nakshatrala, M. K. Mudunuru, and A. J. Valocchi. A numerical framework for diffusion-controlled bimolecular-reactive systems to enforce maximum principles and the non-negative constraint. *Journal of Computational Physics*, 253:278–307, 2013. doi: 10.1016/j.jcp.2013.07.010.

[40] K. B. Nakshatrala, H. Nagarajan, and M. Shabouei. A numerical methodology for enforc-

ing maximum principles and the non-negative constraint for transient diffusion equations. *Communications in Computational Physics*, 19:53–93, 2016. doi: 10.4208/cicp.180615.280815a.

[41] V. I. Naoumov, V. G. Krioukov, A. L. Abdullin, and A. V. Demin. *Chemical Kinetics in Combustion and Reactive Flows: Modeling Tools and Applications*. Cambridge University Press, Cambridge, 2019.

[42] S. Petrovskii and B. Li. An exactly solvable model of population dynamics with density-dependent migrations and the allee effect. *Mathematical Biosciences*, 186(1):79–91, 2003. doi: 10.1016/S0025-5564(03)00098-1.

[43] G. F. Pinder and M. A. Celia. *Subsurface Hydrology*. John Wiley and Sons, Inc., New Jersey, 2006.

[44] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

[45] Z. Ren, B. Wang, and L. Zheng. Numerical analysis on interactions of vortex, shock wave, and exothermal reaction in a supersonic planar shear layer laden with droplets. *Physics of Fluids*, 30(3):036101, 2018. doi: 10.1063/1.5011708.

[46] V. Sabelnikov and V. Vlasenko. *Combustion in Supersonic Flows and Scramjet Combustion Simulation*, pages 585–660. Springer Singapore, Singapore, 2018. doi: 10.1007/978-981-10-7410-3_20.

[47] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003.

[48] A. Sergeev and M. Del Balso. Horovod: Fast and easy distributed deep learning in Tensorflow. *arXiv preprint: 1802.05799*, 2018.

[49] A. Tambue, G. Lord, and S. Geiger. An exponential integrator for advection-dominated reactive transport in heterogeneous porous media. *Journal of Computational Physics*, 229(10): 3957 – 3969, 2010. doi: 10.1016/j.jcp.2010.01.037.

[50] Y. K. Tsang. Predicting the evolution of fast chemical reactions in chaotic flows. *Physical Review E*, 80:026305(8), 2009. doi: 10.1103/PhysRevE.80.026305.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[52] V. V. Vesselinov, M. K. Mudunuru, S. Karra, D. O'Malley, and B. S. Alexandrov. Unsupervised machine learning based on non-negative tensor factorization for analyzing reactive-mixing. *Journal of Computational Physics*, 395:85–104, 2019. doi: 10.1016/j.jcp.2019.05.039.

[53] A. Viguerie, A. Veneziani, G. Lorenzo, D. Baroli, N. Aretz-Nellesen, A. Patton, T. E. Yankeelov, A. Reali, T. J. R. Hughes, and F. Auricchio. Diffusion–reaction compartmental models formulated in a continuum mechanics framework: application to COVID-19, mathematical analysis, and numerical study. *Computational Mechanics*, 66(5):1131–1152, 2020. doi: 10.1007/s00466-020-01888-0.

[54] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. v. Plate, C. Ma, Y. Jernite, J. Plu, T. L. Scao C. Xu, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[55] Y. Xiao, F. Whitaker, T. Xu, and C. Steefel. *Reactive Transport Modeling*. John Wiley and Sons Ltd, New Jersey, 2018.

[56] Y. Xiaofeng, Y. Gui, G. Xiao, Y. Du, L. Liu, and D. Wei. Reacting gas-surface interac-

tion and heat transfer characteristics for high-enthalpy and hypersonic dissociated carbon dioxide flow. *International Journal of Heat and Mass Transfer*, 146:118869, 2020. doi: 10.1016/j.ijheatmasstransfer.2019.118869.

[57] Q. Zhang, S. Hassanizadeh, A. Raoof, M. van Genuchten, and S. Roels. Modeling virus transport and remobilization during transient partially saturated flow. *Vadose Zone Journal*, 11(2):vzj2011–0090, 2012. doi: 10.2136/vzj2011.0090.

[58] C. Zhu and G. Anderson. *Environmental Applications of Geochemical Modeling*. Cambridge University Press, Cambridge, 2002.