PAPER • OPEN ACCESS

Fast modeling and understanding fluid dynamics systems with encoder–decoder networks

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 025022 https://doi.org/10.1088/2632-2153/abd1cf

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
22 June 2020

REVISED
14 October 2020

ACCEPTED FOR PUBLICATION
8 December 2020

PUBLISHED
2 March 2021

PAPER

Fast modeling and understanding fluid dynamics systems with encoder–decoder networks

Rohan Thavarajah, Xiang Zhai, Zheren Ma and David Castineira

QRI AI LLC, Houston, TX, United States of America

E-mail: david.castineira@qrigroup.com

Keywords: fluid dynamics in porous media, spatiotemporal convolutional LSTM, deep learning, surrogate modeling, inverse problems

Abstract

Is a deep learning model capable of representing systems governed by certain first principle physics laws by only observing the system's output? In an effort to simulate two-dimensional subsurface fluid dynamics in porous media, we found that an accurate deep-learning-based proxy model can be taught efficiently by a computationally expensive finite-volume-based simulator. We pose the problem as an image-to-image regression, running the simulator with different input parameters to furnish a synthetic training dataset upon which we fit the deep learning models. Since the data is spatiotemporal, we compare the performance of three alternative treatments of time; a convolutional LSTM, an autoencoder network that treats time as a direct input and an echo state network. Adversarial methods are adopted to address the sharp spatial gradient in the fluid dynamics problem. Compared to traditional simulation, the proposed deep learning approach enables much faster forward computation, which allows us to explore more scenarios with a much larger parameter space given the same time. It is shown that the improved forward computation efficiency is particularly valuable in solving inversion problems, where the physics model has unknown parameters to be determined by history matching. By computing the pixel-level attention of the trained model, we quantify the sensitivity of the deep learning model to key physical parameters and hence demonstrate that the inverse problem can be solved with great acceleration. We assess the efficacy of the machine learning surrogate in terms of its training speed and accuracy. The network can be trained within minutes using limited training data and achieve accuracy that scales desirably with the amount of training data supplied.

1. Introduction

In recent decades, deep learning has demonstrated its power in many different cognitive tasks that were historically believed to be challenging to conceptualize and quantify with mathematical models, such as image recognition, object detection, semantic segmentation, machine translation, and art generation. With the help of massive datasets (Deng *et al* 2009), specially designed network architectures (He *et al* 2015) have been very efficient in learning how to perform complex non-linear feature engineering, capture and master the intrinsically stochastic nature of those tasks, and approximate the human cognitive learning process.

Despite the abundance of success stories in the world of cognitive tasks, the other world of tasks filled with problems and systems that are driven by mathematical logic (i.e. first principle laws, well-defined governing differential equations/models, deterministic or stochastic), still relies on conventional analytical and numerical simulation including finite-difference, finite-volume and finite-element methods. These methods discretize space and time domains into small cells and intervals, transform the partial differential equations into linear and non-linear algebraic problems, and solve those problems numerically. Consider for instance the discrete static 2D Poisson equation which uses simple 3×3 2D kernels $\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$ and $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ to convert the second-order partial differential equation into coupled linear systems that can be solved with $O(n^3)$ time complexity (where $n \times n$ is the size of the discretized 2D grid). In a dynamic system, the time dimension also needs to be discretized and solved sequentially. In computational fluid dynamics (CFD), the

time step must be very small to satisfy the CFL condition (Courant *et al* 1928), making CFD time consuming and numerical errors accumulate with time.

In this paper, we show a novel approach of using a physics-based finite-volume simulator to teach domain-lacking deep learning models to accurately simulate 2D subsurface two-phase fluid dynamics in heterogeneous porous media. The governing equations are Navier–Stokes equations plus Darcy's law

$$\frac{\partial}{\partial t}(\phi\rho_i S_i) - \nabla\cdot(\rho_i\lambda_i\mathbf{K}\cdot\nabla p_i) = q_i^w,\ \ i = o, w, \tag{1}$$

where the subscript $i \in \{o, w\}$ identifies phase as being oil or water, ρ_i is the phase density, λ_i is the relative phase mobility, $\mathbf{K}$ and ϕ are the permeability and porosity distribution of the porous medium, and q_i^w are the sink/source rates at the locations of producing/injecting wells. The phase saturation S_i (i.e. volumetric percentage of the phase) and pressure p_i are the physics quantities we are trying to solve. Since the two phases are directly connected, we will by default choose the water phase pressure and saturation for discussion.

When all parameters are given, the dynamics are determined and can be forward simulated using a finite-volume method. In real-world problems, however, the exact form of the equation is not yet determined due to the uncertainty of subsurface rock properties like permeability $\mathbf{K}$. Meanwhile, the system can be measured/observed at some locations (such as wells). A more important and challenging problem is to, in reverse fashion, estimate the physics parameters $\mathbf{K}$ based on the observed data. In CFD approaches, a forward simulation is conducted with an initial guess of the permeability map and simulated behavior at observable locations is compared with real measurements. The initial guess of permeability is then revised in order to yield new behavior that better aligns with real data. This iterative revision process is called history-matching and is an example of an inverse problem wherein one wishes to infer the exact form of the governing laws based on limited observed data. Forward evaluation is itself expensive, inverse problems, which require many forward evaluations, are much more demanding. We will show that the deep-learning-based surrogate model and its interpretation offer unprecedented advantages to solve the inverse problem.

There have been several recent efforts to capture the physics of a system using fast deep learning surrogates. Examples appear in rainfall prediction, animation, aerodynamics design and of particular interest to us, reservoir simulation (Shi *et al* 2015, Ladický *et al* 2015, Guo *et al* 2016, Zhu and Zabaras 2018, Tompson *et al* 2016). These efforts may be summarized according to two broad strategies.

(a) Data-driven approaches; the network observes the system and derives the physics from data. Our approach is data-driven in nature.
(b) Physics-embedded approaches; a scientist supplies the governing physics equations directly to the network and the network is trained to adhere to them.

Inspired by the success of CNN architectures on image translation problems like pix2pix (Isola *et al* 2016), Zhu and Zabaras (2018) introduce a fully convolutional encoder–decoder network (DenseED) to model the behavior of fluid in heterogeneous media. Encoder–decoder networks are often motivated by an intuition that input and output images share an underlying structure. However, though no such structure is readily apparent, Zhu and Zabaras (2018) find that DenseED successfully translates permeability to steady-state velocity fields. Mo *et al* (2019a) extend DenseED to incorporate GAN loss to tackle highly non-linear outputs (for other examples of GANs applied to proxy particle shower simulations and heat conduction see Paganini *et al* (2018) and Farimani *et al* (2017) respectively).

Time dependence is captured using a variety of strategies. Without altering the architecture of DenseED, Mo *et al* (2019b) model time-dependence by broadcasting time across an additional input channel. By treating time in this manner, Mo *et al* (2019b) are able to furnish predictions at arbitrary time instances. However, fluid flow is highly autoregressive in nature; the state of the system at time t is a function of the system's state at $t-1$ and Mo *et al* (2019b) do not explicitly exploit this structure available in the data. In contrast Wiewel *et al* (2018) combine an encoder–decoder with a long short-term memory network (LSTM) and task the LSTM with learning transitions in the encoder-derived latent space. Shi *et al* (2015) and Wang *et al* (2017) further customize LSTMs for spatiotemporal data by innovating layers that respectively include convolution operations in transition functions and construct a shared memory pool across LSTM layers.

Alternatively, physics-embedded approaches take an unsupervised approach and utilize the governing equations of a system directly. Raissi *et al* (2017a, 2017b) prescribe a framework for embedding physics into a neural network by expressing the loss as the sum of two terms, one that describes the dynamics of the system and another that describes boundary conditions. As Raissi *et al* (2017a) minimize this loss, they find a function approximation that satisfies known dynamics and boundary conditions and so solve the PDE directly. Zhu *et al* (2019) demonstrate how spatial gradients can be computed using Sobel filters and therefore extend the framework pioneered in Raissi *et al* (2017a) to convolutional architectures.

This paper is organized as follows. First, we formulate the problem and describe deep neural network architectures that have successfully learned from a CFD simulator. We compare three different approaches to dealing with time dependence and discuss how accuracy and computation time of the deep learning surrogate scale with more training data. Then we show how training with adversarial loss is helpful for solutions with large spatial gradients. Finally, we demonstrate how model explanation can convert the inverse problem into a gradient-enabled optimization problem that can be solved 4–5 orders of magnitude faster than traditional numerical methods.

2. Method

2.1. Problem formulation

We wish to develop a fast, accurate surrogate to a CFD simulator that solves a collection of PDEs (equation (1)) given a permeability distribution **K** on a uniform 2D Cartesian grid. Since **K** is the input of the system, we represent **K** by 2D tensor $X \in \mathbb{R}^{1\times H\times W}$ where H and W denote the height and width of the grid. In general, we can extend X to 3D and include multiple inputs by expanding the tensor $X \in \mathbb{R}^{n_x\times H\times W\times D}$ where n_x counts the number of inputs and D denotes depth.

The output of the simulator, Y, is spatiotemporal and describes the path of two maps; a saturation map that describes the relative composition of water in each cell, and a pressure map. Again, we represent $Y \in \mathbb{R}^{T\times n_y\times H\times W}$ as a tensor, where n_y counts the total number of output maps ($n_y = 2$) and T is the total number of time steps. The simulator can be represented as a function that maps $f: X \to Y$. We cast the problem as an image-to-image regression treating n_x and n_y as channels and $H \times W$ as the shape of the input and output images.

We nondimensionalize equation (1) and choose $H = W = 50$. Initially, the entire 50 by 50 region is filled with oil, and hence water saturation starts as zero everywhere. An injector placed at the center of the region injects water at a constant rate (source term in equation (1)), replacing and pushing the liquid towards four constant-rate producers at the four corners. This problem setting describes a waterflood, a practice of secondary recovery of crude oil extraction, in which water, less viscous and immiscible with crude oil, is injected into a reservoir to achieve higher long-term ultimate oil recovery as well as maintain subsurface pressure.

A prior permeability map is created using the sequential Gaussian simulation tool provided in SGeMS (Stanford Geostatistical Modeling Software). The tool is based on Kriging and sequentially builds up a variogram from measurements taken at diverse resolutions such that the variogram honors prespecified statistics. The prior permeability map is characterized by diagonal striations that are both observed in practice and likely to generate diverse and therefore interesting behavior across the four producers. We generate 400 new permeability maps by dividing the prior map into six sub-regions based on pixel value and by multiplying each sub-region with six randomly sampled independent multipliers in the range $[e^{-2}, e^{2}]$. After running CFD simulations to time step $T = 30$, we obtain a 400-case dataset filled with synthetic data that corresponds to different physics input (table 1). Three-hundred and thirty-six cases are used as the training set on which we perform 3-fold cross-validation to tune network hyperparameters. The remaining 64 cases are used for testing. The goal is to train networks with the training simulation data that yield reliable predictions of pressure and saturation on the hitherto unseen permeabilities in the test set.

2.2. Architecture

We employ the network architecture developed by Zhu and Zabaras (2018) (DenseED) and incorporate an LSTM as an alternative treatment of time and GAN loss to better simulate shocks. The design presented in Zhu and Zabaras (2018) is a combination of two structures; **convolutional encoder–decoders** and **DenseNet**. Encoder–decoder networks have successfully been applied to image-image translation problems such as image segmentation and pattern infilling (Long *et al* 2014, Badrinarayanan *et al* 2015). Encoder–decoders subject an image to a coarse-refine process. With each successive encoding layer, the network coarsens, extracting higher-level features from the image at lower spatial resolution. Ultimately, the encoder finds an underlying representation of the image in a latent space. With each successive decoding layer, this underlying representation is refined to construct the output.

DenseNets (Huang *et al* 2016) choose convolutions that preserve identical dimensions between inputs and outputs, thereby permitting their concatenation. The input to any one layer is the last layer's output concatenated with all previous inputs. This structure is particularly valuable when analyzing objects across multiple spatial scales. Each successive convolution reflects an expanding receptive field (the region of the input that a pixel in a feature map can 'see'). A 1×1 convolution combines feature maps of varying receptive fields resulting in a network that readily adapts to local versus global phenomena. We present the structure of DenseED in Panel A of figure 1 and detail the specific design of the encoder and decoder in figures 2 and 3.

Table 1. Parameters for ECLIPSE reservoir simulator.

Grid number	$1 \times 50 \times 50$
Grid size, ft	$32.8 \times 32.8 \times 32.8$
Δt, days	3.33
Total days (T)	100
Initial pressure, psi	5863.8
Injector location	(1,25,25)
Injector rate, $\frac{\text{bbl}}{\text{day}}$	2515
Producer locations	(1,1,1), (1,50,1), (1,1,50), (1,50,50)
Producer rate, $\frac{\text{bbl}}{\text{day}}$	628

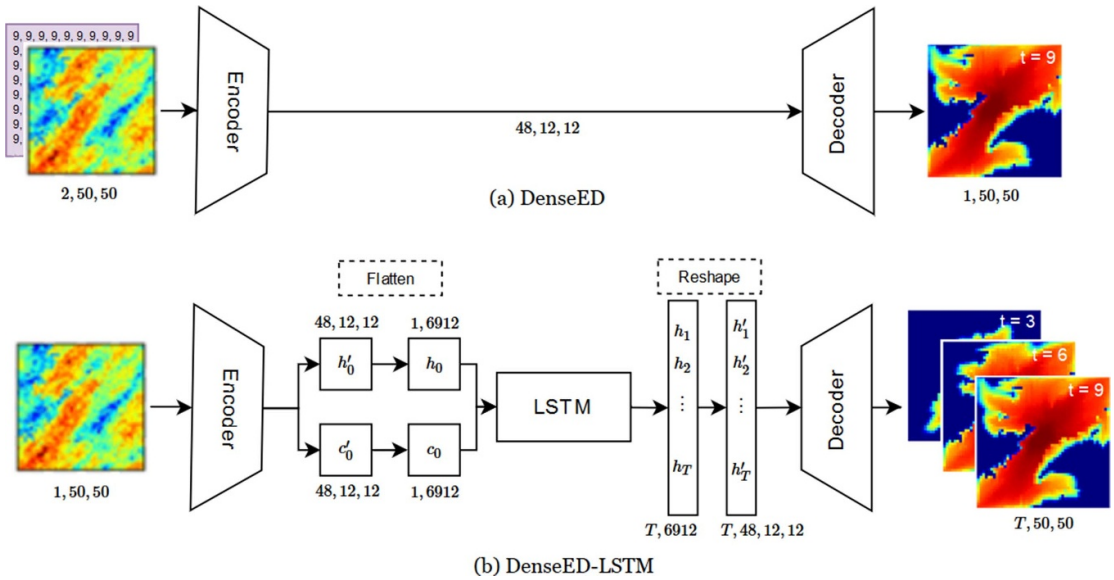

Figure 1. DenseED models time by broadcasting it across a 50×50 input channel (purple) so that it receives two-channel input. DenseED-LSTM receives a single permeability channel as input and outputs T saturation fields. After performing 3-fold cross-validation within the training set, we train both models using Adam as our choice of optimizer (with weight decay $5e^{-4}$ and initial learning rate $3e^{-3}$) and L1 loss over 200 epochs with a batch size of 12. We use PyTorch's default weight initialization and a learning rate scheduler in place of early stopping that reduces the learning rate on a plateau (by a factor of 0.1 if loss fails to improve by a factor of $1e^{-4}$ for 10 epochs).

Dense Encoder		
	INPUT	1x50x50
	CONV2D (k7s2p2)	48x24x24
Encoding blocks (=1)	**DENSEBLOCK (nlayers=3)**	96x24x24
	- for i in range(nlayers)	
	--- bnorm	
	--- relu	
	--- conv2d (k3s1p1, 48+16i, 16)	
	--- concatenate([IN, OUT])	
	DOWN	48x12x12
	- bnorm	
	- relu	
	- conv2d (k1s1p0, 96, 48)	
	- bnorm	
	- relu	
	- conv2d (k3s2p1, 48, 48)	

Figure 2. Encoder design. We represent the arguments of the convolution operations in parentheses. Conv2d (k3s1p1, 48, 16) denotes a 2D convolution with kernel size 3, stride 1, padding 1, which receives 48 features maps as input and outputs 16. DenseED is modularized using encoding and decoding block units which consist of a dense block and a transition block. Each encoding (decoding) block halves (doubles) the dimensions of the output feature map. We find that encoding down to 12×12 features maps works best for saturation and encoding down to 6×6 works best for pressure.

Dense Decoder		
Decoding blocks (=1)	**DENSEBLOCK (nlayers=6)**	144x12x12
	- for i in range(nlayers)	
	--- bnorm	
	--- relu	
	--- conv2d (k3s1p1, 48+16i, 16)	
	--- concatenate([IN, OUT])	
	UP	72x24x24
	- bnorm	
	- relu	
	- conv2d (k1s1p0, 144, 72)	
	- bnorm	
	- relu	
	- convtranspose (k3s2p1p'1, 72, 72)	
	DENSEBLOCK (nlayers=3)	120x24x24
	LASTUP	1x50x50
	- bnorm	
	- relu	
	- conv2d (k1s1p0, 120, 60)	
	- bnorm	
	- relu	
	- convtranspose (k4s2p0p'0, 60, 1)	

Figure 3. Decoder design. See encoder for details.

Mo *et al* (2019b) propose a strategy for modeling time-dependent outputs without altering the design of DenseED by broadcasting time across an input channel. We experiment with incorporating an LSTM (Hochreiter and Schmidhuber 1997) to model transitions in the encoder-derived latent space henceforth referred to as DenseED-LSTM (Panel A of figure 1). We hypothesize that an LSTM, with its memory and transition properties, is particularly apt to model a simulator's dynamic behavior. The persistent cell state will house information about the permeability grid, which is relevant to prediction at every time step. Meanwhile, the hidden state-to-state transitions will receive a strong signal due to the autoregressive nature of the governing physics. Compared to DenseED, the encoder in DenseED-LSTM outputs twice as many feature maps (96 vs. 48). Half of those feature maps are used to initialize the first hidden state and half the first cell state. The LSTM outputs hidden state vectors for T time steps which we decode to predict the saturation path.

To adapt DenseED to our dataset we repeat much of the hyperparameter search presented by Zhu and Zabaras (2018) and detail the best configuration in figure 1. Observing that error is concentrated along the saturation front, we focus and extend our search to design decisions that might mitigate error at the front. Mo *et al* (2019b) also observe similar error characteristics and suggest augmenting the output with a second channel, a binary map that is one if saturation is non-zero and is zero otherwise. Mo *et al* (2019b) propose using a loss function that is a weighted sum of the L1 loss for saturation and cross-entropy for the binary map. Through cross-validation we determine that, irrespective of how we balance the weights, this augmentation strategy fails to improve performance in our case.

We also experiment with coupling pressure and saturation prediction. As we later discuss, we observe that pressure is a more global property than saturation, demanding a larger receptive field. Furthermore, relative to saturation pressure exhibits less variation over time. Owing to these diverging properties, we decouple the problem and separately predict saturation and pressure.

We compare the performance of the DenseED variants to an echo state network. Echo state networks (ESNs) have proven effective in modeling spatiotemporal dynamics of chaotic systems such as synthetic laminar flame behavior and real-world ocean turbulence. Like a recurrent neural network, an ESN also consists of three sets of weights; input $\rightarrow$ hidden, hidden $\rightarrow$ hidden (called the reservoir weights) and hidden $\rightarrow$ output. In contrast to an RNN, the ESN's input and reservoir weight matrix are initialized randomly and are not trainable. This means the entire state vector of the system can be harvested at once and that the output weights can be trained using a simple linear least-squares optimization in a one-shot process. Pathak *et al* (2018) and Zimmermann and Parlitz (2018) scale ESNs to high dimensional spatial data by distributing the prediction task across a large set of parallelizable reservoirs each of which predicts a local region of the system. Heim and Avery (2019) incorporate spatial awareness by replacing the ESN's input weights with a set of functions that extract common image features and by optimizing image Eucledian distance, a metric that is robust to small image perturbations.

Table 2. Model evaluation. Training scores are calculated by refitting each model after having determined the best hyperparameters by 3-fold cross-validation. Training is performed using a Nvidia K80 GPU.

	Train R^2	Train MAE	Test R^2	Test MAE	Time (s)
DenseED	0.991	0.00 763	0.840	0.0412	132
DenseED-LSTM	0.941	0.0185	0.847	0.0381	135
ESN	0.854	0.0602	0.717	0.0898	< 1

We encounter a challenge in that ESNs are not typically applied to problems where initial state is important. The canonical ESN solves a many-to-many sequential task. The network observes a system till time T and learns to propagate the dynamics to yield multi-step forecasts into the future. By the time the system has reached time T, the transient effects of the initial state have washed out and no longer have a large bearing on subsequent forecasts.

However, we have cast our problem as a 1-to-many task. When we deploy the model to a new permeability field, we do not have access to historical saturation data to initialize the ESN. We address this by stacking training observations on top of one another to resemble a single timeseries and include time as an input. Though we are unable to derive a unique initial state for each permeability configuration, our objective is to approximate a mean initial state to use in deployment. We train the ESN on this stacked time series and store the final state at $t = 0$ to serve as the initial state in deployment. Although imperfect we find this approach generates reasonable results. Grid search yields an optimal reservoir size of 250, spectral radius of 0.85, sparsity of 0.4, noise of 0.05 and disables input and teacher shift and scaling.

3. Saturation solutions and alternative treatment of time-dependence

We evaluate the performance of the surrogate in predicting the path of saturation with $R^2 = 1 - \sum_{i=1}^{N}(\hat{y}_i - y_i)^2/(y_0 - \bar{y})^2$, where y_i is a tensor describing the true saturation path for simulation i and $\hat{y}_i$ is the corresponding model prediction. The null model, from which we derive $\bar{y}$, predicts the mean image for all time steps. In table 2, we report the test performance of DenseED and DenseED-LSTM which achieve similar degrees of success. We also report the performance of the ESN which is not as accurate as the DenseED variants likely due to the problem framing mismatch discussed above. However the ESN can be trained quickly in less than a second. So as not to swamp the scales of the axes, results for the ESN have been omitted from some figures. In figure 4, we present predictions for a test permeability map. We observe that error tends to be concentrated around the saturation front for all three models.

To test how well the model generalizes to out-of-distribution permeability maps we experiment with deliberately narrowing the range of multipliers seen in training for Region 1. Recall that logged multipliers are sampled uniformly in the range $[-2, 2]$. We train a model on the 200 cases where logged multipliers for Region 1 are less in absolute value than 1. We test on the 40 observations where the logged multipliers in the region are greater in absolute value than 1.8. Mean absolute error by pixel is plotted in figure 5. We observe that error is concentrated in Region 1 which, despite comprising only 19% of pixels, accounts for 33.7% of the total absolute error in the image. Put differently, the MAE for the entire map is 0.0549, is 0.0972 for Region 1 only and is 0.0449 for the rest of the map.

Had the model learned the underlying physics we would expect it to accurately extrapolate the behavior from other regions to Region 1. In general, from table 2 we conclude that the DenseED variants can be useful proxies for reservoir simulation. But from figure 5 we conclude that their skill is limited to input distributions somewhat similar to those seen in training. It is important to be aware of this limitation and ensure that the model's training set is representative of the task to which it will ultimately be applied.

We further characterize the model's performance by exploring how error varies in time. We expect time will influence error because the difficulty of the prediction task is non-uniform. Initially easy, prediction becomes difficult and then reverts to being easy again over short-, mid- and long-time horizons. The first images are easy to predict because water is initially only concentrated around the injector. As water floods the field, saturation enters a period of rapid flux. In this period, saturation at $t - 1$ bears the least resemblance with its state at t, and prediction is commensurately more difficult. Once saturation steadies, the model can issue a prediction at time t at least as accurately as it did at $t - 1$. Therefore we expect performance to plateau at long time horizons. We observe that for both DenseED and DenseED-LSTM, R^2 follows this trend (figure 6). Performance is highest when t is near zero and deteriorates to a floor. We expect predictions to deteriorate in quality for $t > T(= 30)$ but this is difficult to quantify in our use-case. Any dip in performance from extrapolation is masked by the system reaching an equilibrium that trivializes prediction.

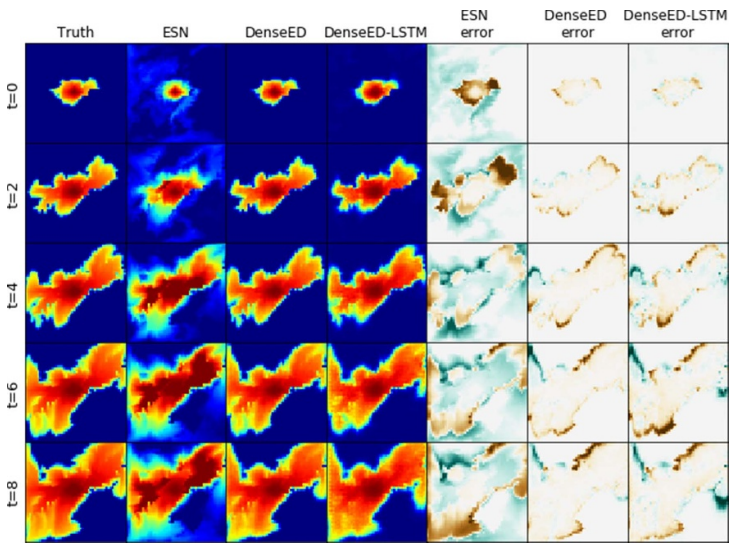

Figure 4. Path of saturation and predictions for a single simulation in the test set. A conventional solver provides ground truth. DenseED, DenseED-LSTM and the ESN are trained using 336 simulations. Saturation is a proportion that takes values between 0 (blue) and 1 (red). We plot error using a diverging color palette and enforce symmetry by setting vmin, vmax $= -0.5$ (brown), 0.5 (blue).

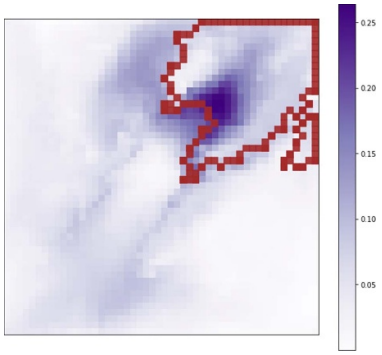

Figure 5. Pixelwise MAE of DenseED-LSTM on out-of-distribution inputs. DenseED and DenseED-LSTM share similar error patterns. Error is highest in Region 1 (red outline) suggesting that the model's ability to extrapolate physics learned in other regions of the map is limited.

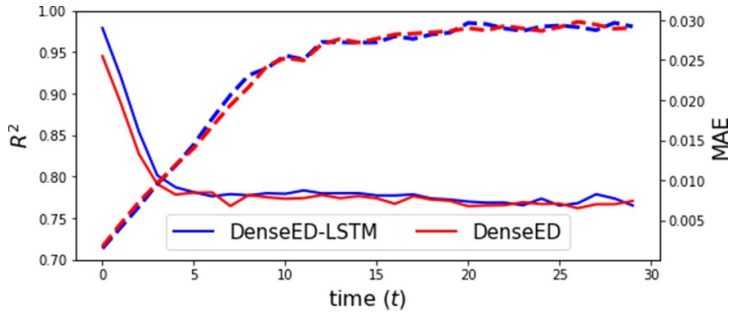

Figure 6. Performance by time step. Test R^2 (left axis, solid lines) and test MAE (right axis, dashed lines). Both DenseED and DenseED-LSTM exhibit initially strong performance that deteriorates to a floor.

Once trained, the neural network can make very fast forward predictions. The CFD simulator took 10 s to finish each simulation when creating the synthetic dataset on an Intel Xeon 2.4GHz 128GB RAM Nvidia K80 workstation. For comparison, the neural network makes batch predictions on 1000 different permeability maps in less than a second on the same machine. The main time overhead of the neural network is in training. Still, it only takes 2 min to train DenseED for 200 epochs to predict a specific time

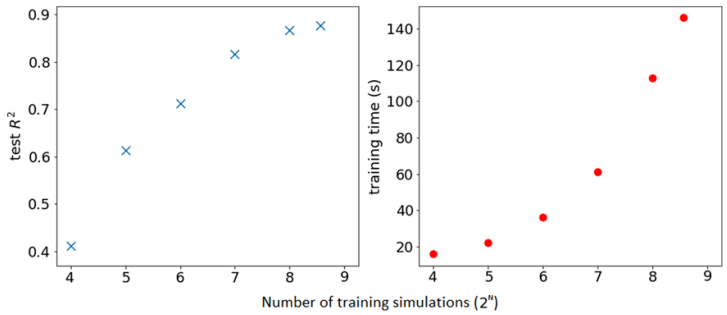

Figure 7. Scaling. DenseED achieves high performance with limited training data.

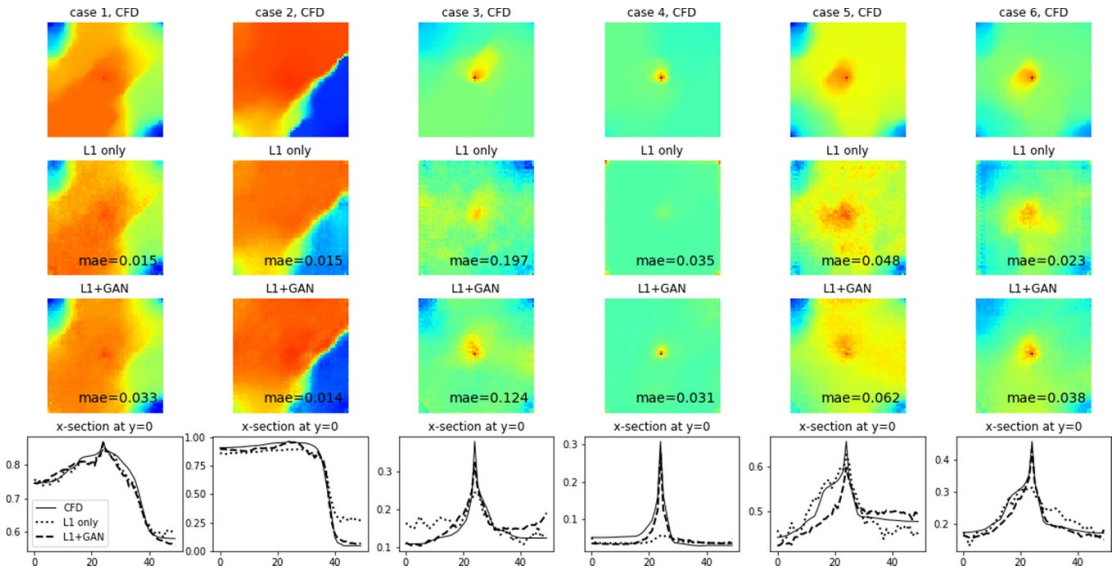

Figure 8. Pressure solutions scaled to $[0, 1]$ at time $t = 25$ of six randomly sampled permeability maps from the test dataset. Top row: True solution from numerical CFD. 2nd row: Outputs from DenseED trained with L1 loss only. 3rd row: Outputs from the same DenseED architecture trained with L1 and GAN loss. Both the L1-only and L1+GAN cases are trained using an initial learning rate of 5×10^{-4} that is halved every 30 epochs over 250 total epochs. The relative weight α for the GAN loss term is 50. The discriminator is trained using soft labels (0.9 and 0.1). The median absolute errors are listed for each of the L1-only and L1+GAN solutions. Bottom row: Cross-sectional view of the three solutions at $y = 0$.

instance with a Nvidia K80. In figure 7, we show how test performance and training time vary with the number of simulations provided in training.

4. Global properties and adversarial effect

Unlike saturation which evolves at the speed of mass motion, pressure is a 'global' physics property established at the speed of sound, several orders of magnitude faster than the fluid's convection speed. Therefore, the pressure solution is affected by the permeability of the entire region. In other words, the receptive field of pressure is global. Indeed, we find that we obtain the best validation results by adding one additional encoding and one additional upsampling block to the DenseED structure, which shrinks the bottleneck of the autoencoder from 12×12 to 6×6. The smaller bottleneck further contains six 3×3-kernel convolutional layers, effectively resulting in a global receptive field of each pixel at the end of the bottleneck.

Fluid dynamic systems are known to have sharp spatial gradient and discontinuity like shocks. For example, the pressure solution scales as $\log(r)$ near the sources and sinks in 2D or cylindrical 3D space. This can lead to enormous spatial gradients close to the wells. Vanilla generative neural networks have difficulty in capturing such behavior because the regular distance-based losses lead to smoother, blurry results in an attempt to resemble many possible solutions. In the case of a fluid dynamic system, although the solution is unique for a given permeability map, we still found large-gradient results are more difficult to generate (see the 2nd row of figure 8).

Generative adversarial networks (GANs) (Goodfellow *et al* 2014) are well known for creating sharp and visually realistic results by introducing a smart loss term learned by comparing the real and generated results. An additional discriminator D is trained simultaneously to penalize the generator if it produces results that the discriminator easily discerns as fake. Similarly to Isola *et al* (2016), we adopt the final objective as

$$G^* = \mathcal{L}(G) + \alpha \cdot \arg\min_G \max_D \mathcal{L}_{GAN}(G, D), \tag{2}$$

where G is the DenseED autoencoder (generator), D is the discriminator which has the same architecture as the encoder of DenseED plus a fully-connected layer to output binary prediction, and $\mathcal{L}$ is the distance-based loss (L1-loss by default) between pixels of generated results and CFD results (true label). The adversarial loss term is given by

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_x[\log(1 - D(G(x))], \tag{3}$$

where x is the permeability map input to DenseED, y is the CFD result, and $G(x)$ is the generated result. In Isola *et al* (2016), a conditional discriminator is utilized to judge if an image is real and also relevant to the input x. In our case, although we do not find the conditional adversarial mechanism enhances the results notably, it does help stabilize the GAN training process. A fundamental difference between equation (3) and regular GAN is the absence of a randomly sampled latent vector. Once the permeability map x is given, the fluid system's dynamics are determined. Therefore, unlike a traditional GAN, we do not feed a random latent vector z as input to the generator. The term α in equation (2) is a weight quantifying the relative importance of GAN loss and is chosen to be 50 so that the GAN loss is about 50% of L1 loss at the conclusion of the training cycle.

Figure 8 compares the pressure solutions from CFD, DenseED trained with L1 loss only and the same DenseED architecture trained with both L1 and GAN loss (equation (3)). GAN loss helps significantly in capturing the large pressure gradient near the central injector. In comparison, L1-only DenseED produces consistently smoother results. Although the overall pixel-wise error of L1-only DenseED is smaller in many cases, the L1+GAN results are scientifically more accurate near the sink/source locations. This is critical because, in the real world, the sink/source locations are where the observable data are sampled.

5. Model explanation and the inverse problem

Interpreting complex machine learning models such as deep neural networks is important because it helps humans understand the behavior of the model, vet whether the prediction is made based on the correct reasoning, and build even better models. Pixel-wise sensitivity and/or spatial attention maps have been used to help understand convolutional neural networks (Bach *et al* 2015, Smilkov *et al* 2017, Lundberg and Lee 2017, Zagoruyko and Komodakis 2016). Given that the deep learning models have been proven capable of accurately mimicking CFD numerical simulations, one important application is to explain the neural network model and uncover its underlying logic.

Considering saturation S as a function of permeability $\mathbf{K} = K(x', y')$ and time t, the 'pixel-wise' sensitivity of saturation to the permeability distribution can be quantified by

$$X(x, y, t, x', y', \mathbf{K}) \equiv \frac{\partial S(x, y, t, \mathbf{K})}{\partial K(x', y')}. \tag{4}$$

The partial in equation (4) represents how much saturation at location and time (x, y, t) would change if the permeability at (x', y') were a bit higher. In real cases, (x, y, t) are the locations/time where the system is observed/measured. Given a fixed location of interest (x_0, y_0) and a time t_0, the local saturation sensitivity to permeability is a 2D slice of equation (4):

$$\mathbf{X}_{x_0, y_0, t_0, \mathbf{K}} \equiv X(x', y')_{x_0, y_0, t_0, \mathbf{K}} = \frac{\partial S(x_0, y_0, t_0, K)}{\partial K(x', y')}. \tag{5}$$

The sensitivity map helps us understand what consequences a slight alteration of the input physics laws may have on the system. However, the sensitivity map $\mathbf{X}$ is extremely difficult to estimate using traditional CFD approaches because there is no explicit formula of $S(x, y, t, \mathbf{K})$. Instead, sensitivity must be calculated by forward computation at very high cost. In the given case, in order to numerically estimate $\mathbf{X}$ on a 50×50 grid at time $t = 25$, 2500 simulations have to be run till this time step for each slightly perturbed input permeability map, which would take 7 h on the same workstation. When the permeability distribution $\mathbf{K} = K(x', y')$ has to be updated iteratively, say, 100 times, in the inversion problem, the CFD-based approach would take more than one month.

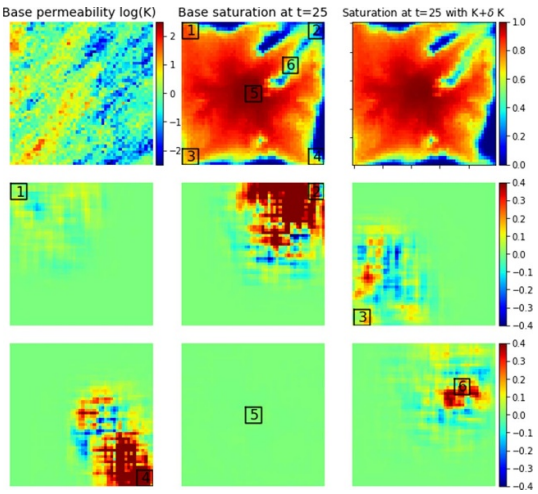

Figure 9. Top rows: Input permeability map (left) and the saturation solution given by DenseED at time $t = 25$ (mid). Top right shows the saturation solution with a slightly perturbed permeability. Rows 2–3: Pixel-wise sensitivity at six observation sites to the input permeability. A value β at coordinate (x, y) in the sensitivity map for Location-**i** means that a Δk increase in the permeability map at (x, y) will result in a $\beta\Delta k$ increase in the saturation at location **i** at time $t = 25$. The partial derivatives are computed by directly backpropagating DenseED and averaged via a 5×5 kernel.

Thanks to the deep-learning surrogate model, $\mathbf{X}$ can be computed directly via backward propagation. Since DenseED and the convLSTM takes $\mathbf{K}$ as input, the derivative of the output of the networks w.r.t. the 'pixels' of the input permeability map is the same as equation (5). We can also compute $\mathbf{X}$ easily via numerical differentiation by running a batch of 2500 forward prediction at low cost. For the same example, evaluating six 2D $\mathbf{X}$ at six different (x_i, y_i, t_i) on the same workstation took only 619 ms, a 40,000-fold acceleration.

Figure 9 shows the saturation sensitivity map at six locations at time $t = 25$ for a given permeability map from the test set. The particular chosen case is more permeable on the left half of the region, resulting in earlier water breakthrough in locations 1 and 3 than locations 2 and 4. As a result, the upper right and lower right corners have higher sensitivity to the permeability distribution between them and the central injector. The central point has zero sensitivity to the permeability map because it is always flooded immediately regardless. Location 6, which is located in a low permeability channel, is most sensitive to its upstream, the small region between itself and the central injector. In some sensitivity maps, certain regions have negative impact, indicating that an increase of permeability in those regions will actually result in a lower saturation; this is because those regions with a higher permeability will redirect some water to other directions and consequently give lower saturation at the locations of interest. The sensitivity maps, effectively obtained by locally linearizing the deep-learning-based surrogate model, honor the underlying physics very well, confirming the deep-learning model's ability to capture the physics by learning from data.

The interpretation of the deep learning model lends itself to a powerful application. In the inverse problem, a dynamic system is governed by certain but not-yet-fully-determined laws. The goal is to find out the exact form of the governing laws so that the resulting dynamics satisfy all the observed data. In our case, the saturation at (x_i, y_i, t_i) is observed and certain; however, the permeability distribution, which gives the exact form of the fluid dynamic equations, is not yet determined. The solution should be an optimal $\mathbf{K}^* = K(x', y')$ so that the resulting S has minimized total discrepancy with all the observed data, i.e.

$$\mathbf{K}^* = \underset{\mathbf{K}}{\operatorname{argmin}} \mathcal{L}_{\mathrm{IP}}, \quad \mathcal{L}_{\mathrm{IP}} = \Sigma_i ||S(x_i, y_i, t_i, \mathbf{K}) - S_i|| \tag{6}$$

where $\{S_i\}$ are the set of real measurements at locations/times (x_i, y_i, t_i). With L2 norm, the loss of the inverse problem is differentiable w.r.t. $\mathbf{K}$:

$$\nabla_{\mathbf{K}} \mathcal{L}_{\mathrm{IP}} = 2\Sigma_i (S(x_i, y_i, t_i, \mathbf{K}) - S_i) \cdot \mathbf{X}_{x_i, y_i, t_i, \mathbf{K}}. \tag{7}$$

Given that $\mathbf{X}$ can be readily evaluated with the deep-learning surrogate model, the above problem can be solved using gradient-based optimization methods iteratively via

$$\mathbf{K} \leftarrow \mathbf{K} - \gamma \cdot \nabla_{\mathbf{K}} \mathcal{L}_{\mathrm{IP}},$$

where γ is the learning rate in the above gradient-descent iteration. In other words, in every iteration of the inversion optimization, the input permeability map is added/subtracted by a linear combination of the

sensitivity maps shown in figure 9, and then the sensitivity maps are recomputed to adjust for the change of the input permeability distribution. For instance, we perturbed the input permeability by adding 0.1 times the sensitivity map 2 and subtracting 0.3 times the sensitivity map 3, to obtain a desired saturation map (upper right corner of figure 9) in which location 2 has higher saturation and location 4 has lower saturation.

We show in this section that model explanation offers an unprecedented advantage in solving the inverse problem, the true challenge that is not yet achievable with the traditional CFD method due to the limitation of computation power.

6. Summary and discussion

We have presented a novel method for modeling a special fluid dynamic system with deep learning. We show that, by sampling a representative distribution of the exact form of the physics laws, it is possible to teach a neural network to accurately simulate the dynamic system, and make predictions at high accuracy with new physics not explicitly provided in the training data; moreover, the deep learning approach provides a significantly faster way to forward evaluate the system, potentially helping solve the motivating inverse problem. The non-linear partial dependence of the system behavior to the imposed physics can be quantified using the techniques of neural network explanation. This provides a much easier way to solve the inverse problem, which would be otherwise infeasible using traditional numerical methods due to their high cost.

Data availability statement

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

ORCID iD

Rohan Thavarajah https://orcid.org/0000-0002-0360-602X

References

Bach S Binder A Montavon G Klauschen F Müller K-R and Samek W 2015 On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation *PloS One* **10** 1–46

Badrinarayanan V Kendall A and Cipolla R 2015 SegNet: a deep convolutional encoder–decoder architecture for image segmentation (arXiv: 1511.00561) [cs]

Courant R Friedrichs K and Lewy H 1928 Über die partiellen differenzengleichungen der mathematischen physik *Math Ann* **100** 32–74

Deng J Dong W Socher R Li L-J Li K and Fei-Fei L 2009 ImageNet: A Large-Scale Hierarchical Image Database *Cvpr09*

Farimani A B Gomes J and Pande V S 2017 Deep learning the physics of transport phenomena (arXiv: 1709.02432) [physics]

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets *Advances in Neural Information Processing Systems 27*, eds Z Ghahramani, M Welling, C Cortes, N D Lawrence and K Q Weinberger (New York: Curran Associates, Inc.) pp 2672–80

Guo X, Li W and Iorio F 2016 Convolutional neural networks for steady flow approximation *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, KDD'16* (*New York, NY, USA*: ACM) pp 481–90

He K Zhang X Ren S and Sun J 2015 Deep residual learning for image recognition *CoRR* abs/1512.03385

Heim N and Avery J E 2019 Adaptive anomaly detection in chaotic time series with a spatially aware echo state network (arXiv: 1909.01709) [cs, stat]

Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80

Huang G Liu Z van der Maaten L and Weinberger K Q 2016 Densely Connected Convolutional Networks (arXiv: L1608.06993) [cs]

Isola P Zhu J-Y Zhou T and Efros A A 2016 Image-to-Image Translation with Conditional Adversarial Networks (arXiv:1611.07004) [cs]

Ladický L Jeong S Solenthaler B Pollefeys M and Gross M 2015 Data-driven fluid simulations using regression forests *ACM Trans. Graph.* **34** 199:1–199:9

Long J, Shelhamer E and Darrell T 2014 Fully convolutional networks for semantic segmentation *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA: Curran Associates, Inc.) pp 3431–40

Lundberg S M and Lee S-I 2017 A unified approach to interpreting model predictions *Advances in Neural Information Processing Systems 30*, eds I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan and R Garnett (New York: Curran Associates, Inc.) pp 4765–74

Mo S Zabaras N Shi X and Wu J 2019a Integration of adversarial autoencoders with residual dense convolutional networks for inversion of solute transport in non-Gaussian conductivity fields (arXiv: 1906.11828) [physics, stat]

Mo S Zhu Y Zabaras N Shi X and Wu J 2019b Deep convolutional encoder–decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media *Water Resour. Res.* **55** 703–28

Paganini M de Oliveira L and Nachman B 2018 Accelerating science with generative adversarial networks: an application to 3D particle showers in multi-layer calorimeters *Phys. Rev. Lett.* **120** 042003

Pathak J Hunt B Girvan M Lu Z and Ott E 2018 Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach *Phys. Rev. Lett.* 120

Raissi M Perdikaris P and Karniadakis G E 2017a Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations (arXiv: 1711.10561) [cs, math, stat]

Raissi M Perdikaris P and Karniadakis G E 2017b Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations (arXiv: 1711.10566) [cs, math, stat]

Shi X, Chen Z, Wang H, Yeung D-Y, Wong W-k and Woo W-c 2015 Convolutional LSTM network: a machine learning approach For precipitation nowcasting p 9 (arXiv:1506.04214)
Smilkov D Thorat N Kim B Viégas F B and Wattenberg M 2017 Smoothgrad: removing noise by adding noise *CoRR* abs/1706.03825
Tompson J Schlachter K Sprechmann P and Perlin K 2016 Accelerating Eulerian fluid simulation with convolutional networks (arXiv: 1607.03597) [cs]
Wang Y, Long M, Wang J, Gao Z and Yu P S 2017 PredRNN: recurrent neural networks for predictive learning using spatiotemporal LSTMs Advances in Neural Information Processing Systems 30 (New York: Curran Associates, Inc.) pp 879–88
Wiewel S Becher M and Thuerey N 2018 Latent-space physics: towards learning the temporal evolution of fluid flow (arXiv: 1802.10123) [cs]
Zagoruyko S and Komodakis N 2016 Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer *CoRR* abs/1612.03928
Zhu Y and Zabaras N 2018 Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification *J. Comput. Phys.* **366** 415–47
Zhu Y Zabaras N Koutsourelakis P-S and Perdikaris P 2019 Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data *J. Comput. Phys.* **394** 56–81
Zimmermann R S and Parlitz U 2018 Observing spatio-temporal dynamics of excitable media using reservoir computing *Chaos* **28** 043118