Deep learning of turbulent scalar mixing

Maziar Raissi
Division of Applied Mathematics, Brown University, Providence, Rhode Island 02912, USA

Hessam Babaee and Peyman Givi
Department of Mechanical Engineering and Materials Science, University of Pittsburgh, Pittsburgh, Pennsylvania 15261, USA

(Received 15 November 2018; published 2 December 2019)

Based on recent developments in *physics-informed deep learning* and *deep hidden physics models*, we put forth a framework for discovering turbulence models from scattered and potentially noisy spatiotemporal measurements of the probability density function (PDF). The models are for the conditional expected diffusion and the conditional expected dissipation of a Fickian scalar described by its transported single-point PDF equation. The discovered models are appraised against an exact solution derived by the amplitude mapping closure (AMC)–Johnson-Edgeworth translation (JET) model of binary scalar mixing in homogeneous turbulence.

DOI: 10.1103/PhysRevFluids.4.124501

I. INTRODUCTION

The problem of turbulent scalar mixing has been the subject of widespread investigation for several decades now [1–4]. The problem is explicitly exhibited in the transported probability density function (PDF) description of turbulence in Reynolds-averaged Navier-Stokes (RANS) simulations. With the single-point PDF descriptor, the effects of mixing of a Fickian scalar appear in unclosed forms of the conditional expected dissipation and/or the conditional expected diffusion terms [3]. A similar closure problem is encountered in large eddy simulation (LES) via the probabilistic filtered density function (FDF) [5]. Development of closures for these terms has been, and continues to be, an area of active research; see, e.g., Refs. [4,6–9] for reviews. The overarching goal of turbulence modeling is to find accurate closures for the unclosed terms that appear in PDF/FDF transport equations. As a common practice in turbulence modeling, the unclosed terms are formulated versus closed terms. The form of this closure is based on physical inspection of the problem at hand and it is the major source of modeling uncertainty in turbulence closure.

In this paper, we introduce a paradigm for turbulent scalar mixing closure in which the unclosed terms are learned from high-fidelity observations. Such observations may come from direct numerical simulation (DNS), [10–12] or space-time resolved experimental measurements, e.g., [13–15]. Obviously, in DNS, the unclosed term can be extracted directly from the simulated results. However, for most realistic applications, performing DNS is cost prohibitive. On the other hand, finding closure from experimental data involves taking derivatives in space-time and decomposition space from the experimental data (in some cases high-order derivatives), which is nontrivial and, even if possible, introduces new uncertainty on the closure depending on the space-time resolution of the measurements. Our ultimate goal is to develop a closure discovery framework that learns the closure from sparse high-fidelity data, such as experimental measurements. The proposed framework replaces the *guessing* work often involved in such model development with a data-driven approach that uncovers the closure from data in a systematic fashion. Our approach draws inspiration from the early and contemporary contributions in deep learning for partial differential equations [16–22] and data-driven modeling strategies [23–25], and particularly relies on recent developments in *physics-informed deep learning* [26] and *deep hidden physics models* [27].

The means for exploiting structured information for construction of data-efficient and physics-informed learning machines are demonstrated in Refs. [28–30]. In these studies, the Gaussian process regression [31] is employed to devise functional representations that are tailored to a given linear operator to provide uncertainty estimates for several prototype problems. Extensions to nonlinear problems are proposed in Refs. [32,33] in the context of both inference and system identification. However, the Bayesian nature of Gaussian process regression requires certain prior assumptions that may limit the representation capacity of the model and give rise to robustness or brittleness issues, especially for nonlinear problems [34,35]. Physics-informed deep learning [36–39] takes a different approach by employing deep neural networks and leveraging their well-known capability as universal function approximators. In this setting, one can directly tackle nonlinear problems without the need for committing to any prior assumptions, linearization, or local time-stepping.

Several examples of machine learning approaches for predictive modeling of physical systems are given in Refs. [40–52]. All these approaches employ machine learning algorithms such as support vector machines, random forests, Gaussian processes, and feed-forward, convolutional, recurrent neural networks merely as *black-box* tools. Our goal here is to open the black box, understand the mechanisms inside it, and utilize them to develop new methods and tools that could potentially lead to new models, novel regularization procedures, and efficient and robust inference techniques. In particular, opening the black box involves understanding and appreciating the key role played by automatic differentiation within the deep learning field. To this end, the proposed work draws inspiration from the early contributions in Refs. [16,17,53], as well as the contemporary works in Refs. [54–57].

The focus of this paper is a turbulent scalar mixing closure in which the unclosed terms are learned from high-fidelity observations. As a demonstration example, we consider the binary scalar mixing which has been very useful for PDF closure developments [58–68]. The problem is typically considered in the setting of a spatially homogeneous flow in which the temporal transport of the scalar PDF is considered. In this setting, development of a closure which can accurately predict the evolution of the PDF is of primary importance. The relative simplicity of the problem makes it well suited for both DNS and laboratory experiments. The literature is rich with a wealth of data obtained by these means; see, e.g., Refs. [10–12,69–76]. We will demonstrate that our proposed framework rediscovers the conditional expected dissipation and diffusion.

II. BINARY SCALAR MIXING

We consider the mixing of a Fickian passive scalar $\psi = \psi(t, \boldsymbol{x})$ (t denotes time and $\boldsymbol{x}$ is the position vector) with diffusion coefficient Γ from an initially symmetric binary state within the bounds $-1 \leqslant \psi \leqslant +1$. Therefore, the single-point PDF of ψ at the initial time is $P(0, \psi) = \frac{1}{2}[\delta(\psi - 1) + \delta(\psi + 1)]$, where ψ denotes the composition sample space for $\psi(t, \boldsymbol{x})$. Thus $\langle \psi \rangle(t = 0) = 0$, $\sigma^2(0) = 1$, where $\langle\ \rangle$ indicates the probability mean (average) and σ^2 denotes the variance. In homogeneous turbulence, the PDF transport is governed by

$$\frac{\partial P}{\partial t} + \frac{\partial^2(\mathcal{E}P)}{\partial \psi^2} = 0, \quad -1 \leqslant \psi \leqslant +1, \tag{1}$$

where $\mathcal{E}$ represents the expected value of the scalar dissipation $\xi = \Gamma \nabla \psi \cdot \nabla \psi$, conditioned on the value of the scalar

$$\mathcal{E} = \mathcal{E}(t, \psi) = \langle \xi | \psi(t, \boldsymbol{x}) = \psi \rangle, \tag{2}$$

where the vertical bar denotes the conditional value. Equation (1) is also expressed by

$$\frac{\partial P}{\partial t} + \frac{\partial (DP)}{\partial \psi} = 0, \tag{3}$$

where D denotes the conditional expected diffusion

$$D = D(t, \psi) = \langle \Gamma \nabla^2 \psi | \psi(t, \boldsymbol{x}) = \psi \rangle. \tag{4}$$

The closure problem in the PDF transport is associated with the unknown conditional expected dissipation $\mathcal{E}$ and/or the conditional expected diffusion D. At the single-point level none of these conditional averages are known and neither are their unconditional (total) mean values:

$$\epsilon(t) = \int_{-1}^{+1} P(t, \psi)\mathcal{E}(t, \psi) d\psi = -\int_{-1}^{+1} \psi P(t, \psi) D(t, \psi) d\psi. \tag{5}$$

III. DEEP LEARNING SOLUTION

Given data $\{t^n, \psi^n, P^n\}_{n=1}^N$ on the PDF $P(t, \psi)$, we are interested in inferring the unknown dissipation $\mathcal{E}(t, \psi)$ and diffusion $D(t, \psi)$ by leveraging Eqs. (1) and (3), respectively, and consequently solving the closure problem. The data may be obtained from DNS or experimental measurements.

A. Conditional expected diffusion

Inspired by recent developments in *physics-informed deep learning* [26] and *deep hidden physics models* [27], we propose to approximate the function

$$(t, \psi) \longmapsto (P, D)$$

by a deep neural network taking as inputs t and ψ while outputting P and D. This choice is motivated by modern techniques for solving forward and inverse problems involving partial differential equations, where the unknown solution is approximated either by a Gaussian process [28,29,32,33,77–80] or a neural network [26,27,81–84]. Moreover, placing a prior on the solution itself is fully justified by the similar approach pursued in the past century by classical methods of solving partial differential equations such as finite element, finite difference, or spectral methods, where one would expand the unknown solution in terms of an appropriate set of basis functions. However, the classical methods suffer from the curse of dimensionality, mainly due to their reliance on spatiotemporal grids. In contrast, modern techniques avoid the tyranny of mesh generation, and consequently the curse of dimensionality [19,82], by approximating the unknown solution with a neural network [36–38] or a Gaussian process. This transforms the problem of solving a partial differential equation into an optimization problem. This is enabling as it allows us to solve forward, backward (inverse), data-assimilation, data-driven discovery, and control problems (in addition to many other classes of problems of practical interest) using a unified framework. On the flip side of the coin, this can help us design physics-informed learning machines.

The aforementioned prior assumption along with Eq. (3) will allow us to obtain the following *physics-informed neural network* (see Fig. 1):

$$R := \frac{\partial P}{\partial t} + \frac{\partial (DP)}{\partial \psi}.$$

We obtain the required derivatives to compute the residual network $R(t, \psi)$ by applying the chain rule for differentiating compositions of functions using automatic differentiation [85]. It is worth emphasizing that automatic differentiation is different from, and in several respects superior to, numerical or symbolic differentiation, two commonly encountered techniques of computing derivatives. In its most basic description [85], automatic differentiation relies on the fact that all numerical computations are ultimately compositions of a finite set of elementary operations for which derivatives are known. Combining the derivatives of the constituent operations through

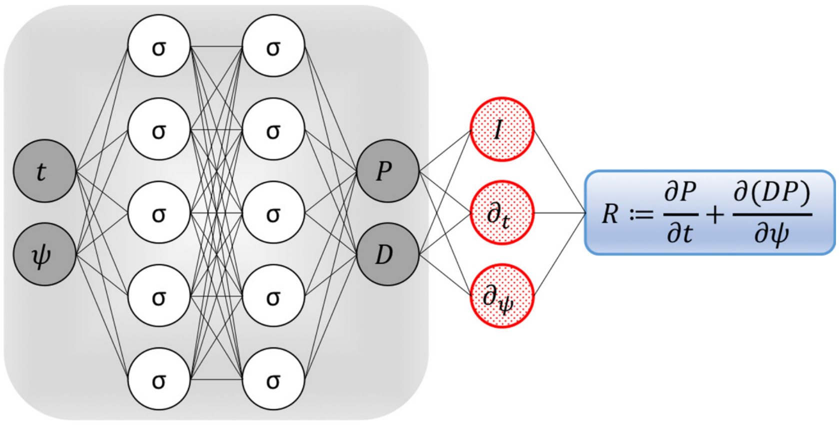

σ
t
ψ
P
D
I
∂_t
∂_ψ
$R := \frac{\partial P}{\partial t} + \frac{\partial (DP)}{\partial \psi}$

FIG. 1. *Conditional expected diffusion network:* A plain vanilla, densely connected (physics-uninformed) neural network with 10 hidden layers and 50 neurons per hidden layer per output variable (i.e., $2 \times 50 = 100$ neurons per hidden layer) takes the input variables t and ψ while outputting P and D. As for the activation functions, we use $\sigma(x) = x$ sigmoid(x) known in the literature as *Swish*. For illustration purposes only, the network depicted in this figure comprises two hidden layers and five neurons per set of hidden layers. We employ automatic differentiation to obtain the required derivatives to compute the residual (physics-informed) network R. The total loss function is composed of the regression loss of the probability density function P and the loss imposed by the partial differential equation R. Here, I denotes the identity operator, and the differential operators ∂_t and ∂_ψ are computed using automatic differentiation and can be thought of as "activation operators." Moreover, the gradients of the loss function are back-propagated through the entire network to train the neural network parameters using the Adam optimizer.

the chain rule gives the derivative of the overall composition. This allows accurate evaluation of derivatives at machine precision with ideal asymptotic efficiency and only a small constant factor of overhead. In particular, to compute the required derivatives we rely on TENSORFLOW [86], which is a popular and relatively well documented open source software library for automatic differentiation and deep learning computations.

Parameters of the neural networks $P(t, \psi)$ and $D(t, \psi)$ can be learned by minimizing the following loss function:

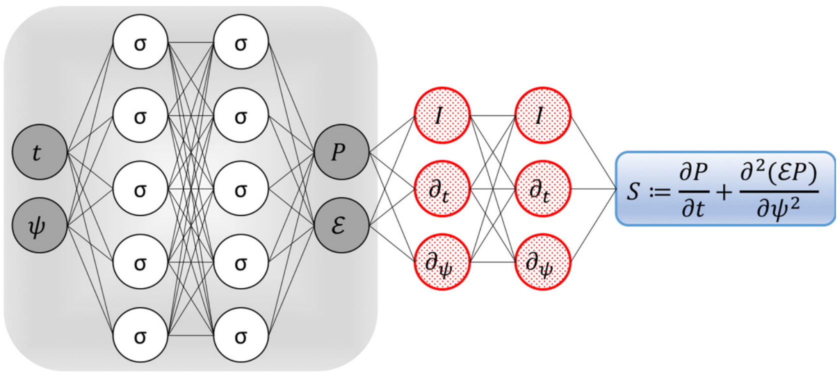
$$\sum_{n=1}^{N} |P(t^n, \psi^n) - P^n|^2 + \sum_{n=1}^{N} |R(t^n, \psi^n)|^2,$$

where $\{t^n, \psi^n, P^n\}_{n=1}^{N}$ represents the data on the probability density function $P(t, \psi)$. Here, the first summation corresponds to the training data on the probability density function $P(t, \psi)$ while the second summation enforces the structure imposed by Eq. (3) at a finite set of measurement points whose number and locations are taken to be the same as the training data. However, it should be pointed out that the number and locations of the points on which we enforce the set of partial differential equations could be different from the actual training data. Although not pursued in the current work, this could significantly reduce the required number of training data on the probability density function.

B. Conditional expected dissipation

Alternatively, one could proceed by approximating the function

$$(t, \psi) \longmapsto (P, \mathcal{E})$$



σ
t
ψ
P
$\mathcal{E}$
I
∂_t
∂_ψ
$S := \frac{\partial P}{\partial t} + \frac{\partial^2(\mathcal{E}P)}{\partial \psi^2}$

FIG. 2. *Conditional expected dissipation network:* A plain vanilla, densely connected (physics-uninformed) neural network, with 10 hidden layers and 50 neurons per hidden layer per output variable (i.e., $2 \times 50 = 100$ neurons per hidden layer), takes the input variables t and ψ while outputting P and $\mathcal{E}$. As for the activation functions, we use $\sigma(x) = x$ sigmoid(x), known in the literature as *Swish*. For illustration purposes only, the network depicted in this figure comprises two hidden layers and five neurons per set of hidden layers. We employ automatic differentiation to obtain the required derivatives to compute the residual (physics-informed) network S. If a term does not appear in the blue boxes (e.g., $\frac{\partial^2 P}{\partial t^2}$ or $\frac{\partial^2 P}{\partial t \partial \psi}$), its coefficient is assumed to be zero. It is worth emphasizing that unless the coefficient in front of a term is nonzero, that term is not going to appear in the actual "compiled" computational graph and is not going to contribute to the computational cost of a feed-forward evaluation of the resulting network. The total loss function is composed of the regression loss of the probability density function P and the loss imposed by the differential equation S. Here, I denotes the identity operator, and the differential operators ∂_t and ∂_ψ are computed using automatic differentiation and can be thought of as "activation operators." Moreover, the gradients of the loss function are back-propagated through the entire network to train the neural network parameters using the Adam optimizer.

by a deep neural network taking as inputs t and ψ while outputting P and $\mathcal{E}$. This prior assumption along with Eq. (1) will allow us to obtain the following *physics-informed neural network* (see Fig. 2):

$$S := \frac{\partial P}{\partial t} + \frac{\partial^2(\mathcal{E}P)}{\partial \psi^2}.$$

We use automatic differentiation [85] to acquire the required derivatives to compute the residual network $S(t, \psi)$. Parameters of the neural networks $P(t, \psi)$ and $\mathcal{E}(t, \psi)$ can be learned by minimizing the following loss function:

$$\sum_{n=1}^{N} |P(t^n, \psi^n) - P^n|^2 + \sum_{n=1}^{N} |S(t^n, \psi^n)|^2,$$

where $\{t^n, \psi^n, P^n\}_{n=1}^N$ represents the data on the probability density function $P(t, \psi)$. Here, the first summation corresponds to the training data on the probability density function $P(t, \psi)$ while the second summation enforces the structure imposed by Eq. (1) at a finite set of measurement points whose number and locations are taken to be the same as the training data.

IV. ASSESSMENT

To assess the performance of our deep learning algorithms, we consider the amplitude mapping closure (AMC) [87–89]. This provides the external closure for the PDF transport in an implicit manner. As explained in detail by Pope [89], the AMC process involves a *mapping* of the random field of interest ψ to a stationary Gaussian reference field ψ_0 via a transformation $\psi = \chi(\psi_0, t)$. Once this relation is established, the PDF of the random variable ψ, $P(t, \psi)$, is related to that of a

Gaussian distribution. In a domain with fixed upper and lower bounds, the corresponding form of the mapping function is obtained by Pope [89]. The solution for a symmetric field, here with zero mean, is represented in terms of an unknown time τ:

$$\chi(\psi_0, \tau) = \operatorname{erf}\left(\frac{\psi_0}{\sqrt{2}G}\right), \quad G(\tau) = \sqrt{\exp(2\tau) - 1}. \tag{6}$$

With this transformation, the PDF is determined from the physical requirement

$$P[\chi(\psi_0, \tau), \tau]d\chi = P_G(\psi_0)d\psi_0, \tag{7}$$

where P_G denotes the PDF of a standardized Gaussian distribution, i.e., $P_G(\psi_0) = \frac{1}{\sqrt{2\pi}}\exp(-\psi_0^2/2)$. A combination of Eqs. (6) and (7) yields

$$P(\tau, \psi) = \frac{G}{2}\exp\{-(G^2 - 1)[\operatorname{erf}^{-1}(\psi)]^2\}. \tag{8}$$

The AMC captures many of the basic features of the binary mixing problem, namely, the inverse diffusion of the PDF in the composition domain from a double δ distribution to an asymptotic approximate Gaussian distribution centered around $\langle\psi\rangle$ as the variance goes to zero (or $G \to \infty$). There are other means of "driving" the PDF toward Gaussianity (or any other distribution) in a physically acceptable manner. The Johnson-Edgeworth translation (JET) [90] involves the transformation of the random physical field ψ to a fixed standard Gaussian (or any other) reference

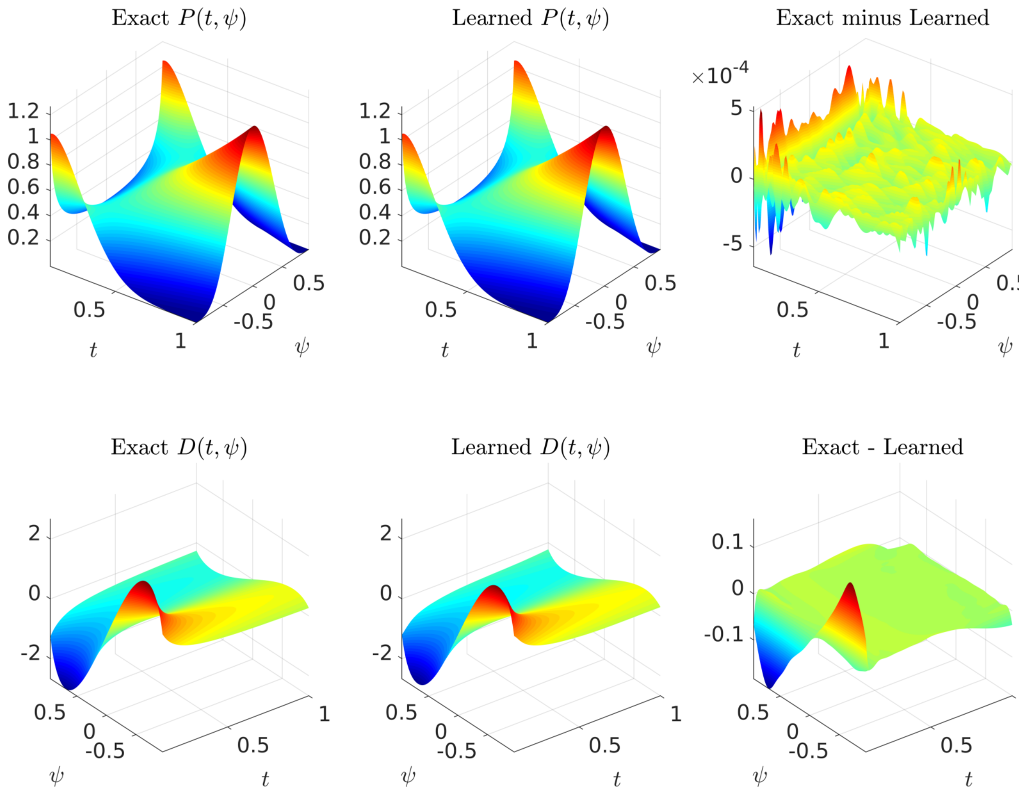


FIG. 3. *Conditional expected diffusion:* The exact probability density function $P(t, \psi)$ alongside the learned one is depicted in the top panels, while the exact and learned conditional expected diffusion $D(t, \psi)$ are plotted in the bottom panels. It is worth highlighting that the algorithm has seen no data whatsoever on the diffusion coefficient.

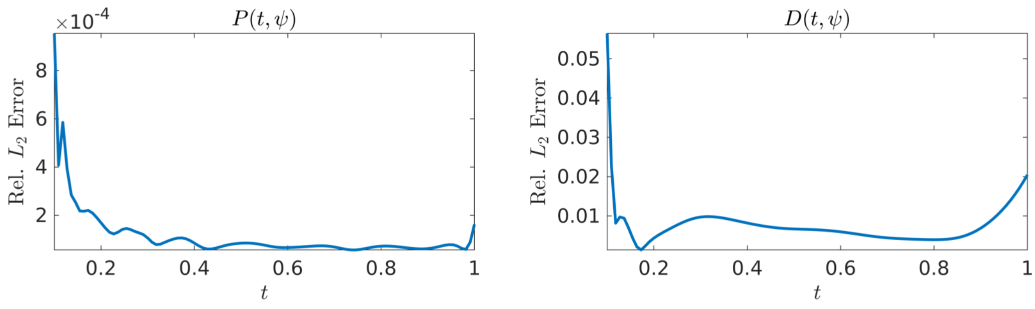


FIG. 4. The relative L_2 error as a function of time for probability density function $P(t, \psi)$ (left) and conditional expected diffusion $D(t, \psi)$ (right).

field by means of a translation of the form

$$\psi = \mathcal{Z}\left[\frac{\psi_0}{\gamma(t)}\right], \quad \gamma(t=0) = 0 \leqslant \gamma(t) \leqslant \gamma(t \to \infty) \to \infty.$$

The function $\gamma(t)$ here plays a role similar to that of G in the AMC. With appropriate form for the function $\mathcal{Z}$, the scalar PDF is determined. In this manner, many frequencies can be generated. The AMC, for example, is recovered by the translation $\mathcal{Z} = \text{erf}(\psi_0/\gamma)$ and so can also be labeled as the erf^{-1}-normal distribution. Recognizing this translation, the relation between τ and the physical time t can be determined through knowledge of the higher-order statistics. For example, the normalized variance

$$\frac{\langle\sigma^2\rangle(\tau)}{\langle\sigma^2\rangle(0)} = \frac{2}{\pi}\arctan\left(\frac{1}{G\sqrt{G^2+2}}\right) \tag{9}$$

determines t through specification of the total mean dissipation $\epsilon(t) = -\sigma\frac{d\sigma}{dt}$. With the knowledge of this dissipation, all of the conditional statistics are determined [90,91]:

$$\frac{\mathcal{E}(t,\psi)}{\epsilon(t)} = \left(\sqrt{\frac{1+\sin\left[\frac{\pi\sigma^2(t)}{2\sigma^2(0)}\right]}{1-\sin\left[\frac{\pi\sigma^2(t)}{2\sigma^2(0)}\right]}}\right)\exp\{-2[\text{erf}^{-1}(\psi)]^2\}, \tag{10}$$

$$\frac{D(t,\psi)}{\epsilon(t)} = \left(\frac{-\sqrt{\pi}}{\sin\left[\frac{\pi\sigma^2(t)}{2\sigma^2(0)}\right]}\sqrt{\frac{1+\sin\left[\frac{\pi\sigma^2(t)}{2\sigma^2(0)}\right]}{1-\sin\left[\frac{\pi\sigma^2(t)}{2\sigma^2(0)}\right]}}\right)\exp\{-[\text{erf}^{-1}(\psi)]^2\}\text{erf}^{-1}(\psi). \tag{11}$$

V. RESULTS

In the following, the AMC (or the erf^{-1}-normal distribution) is utilized to assess the performance of our deep learning framework. In particular, Fig. 3 depicts the exact and the learned conditional expected diffusion $D(t, \psi)$. It is worth highlighting that the algorithm has seen no data whatsoever on the conditional expected diffusion. To obtain the results reported in this figure we are approximating $P(t, \psi)$ and $D(t, \psi)$ by a deep neural network consisting of 10 hidden layers with 100 neurons

TABLE I. *Conditional expected diffusion:* Relative L_2 error for P and D vs number of training points.

Number of training points	20000	10000	5000	2500
Relative L_2 error for P	1.63×10^{-4}	1.70×10^{-4}	1.54×10^{-4}	2.08×10^{-4}
Relative L_2 error for D	9.84×10^{-2}	3.27×10^{-2}	5.60×10^{-2}	3.97×10^{-2}

TABLE II. *Conditional expected diffusion:* Relative L_2 error for P and D vs levels of noise.

Noise level (σ)	0.1	0.05	0.01
Relative L_2 error for P	5.02×10^{-3}	1.98×10^{-3}	3.12×10^{-4}
Relative L_2 error for D	8.73×10^{-2}	3.63×10^{-2}	9.23×10^{-2}

per each hidden layer (see Fig. 1). As for the activation functions, we use x sigmoid(x), known in the literature [92] as the *Swish* activation function. The smoothness of Swish and its similarity to a rectified linear unit or ReLU make it a suitable candidate for an activation function while working with physics-informed neural networks [27]. In general, the choice of a neural network's architecture (e.g., number of layers or neurons and form of activation functions) is crucial and in many cases still remains an art that relies on one's ability to balance the tradeoff between *expressivity* and *trainability* of the neural network [93]. Our empirical findings so far indicate that deeper and wider networks are usually more expressive (i.e., they can capture a larger class of functions) but are often more costly to train (i.e., a feed-forward evaluation of the neural network takes more time, and the optimizer requires more iterations to converge). In this work, we have tried to choose the neural networks' architectures in a consistent fashion throughout the manuscript by setting the number of hidden layers to 10 and the number of neurons to 50 per output variable. Consequently, there might exist other architectures that improve some of the results reported here.

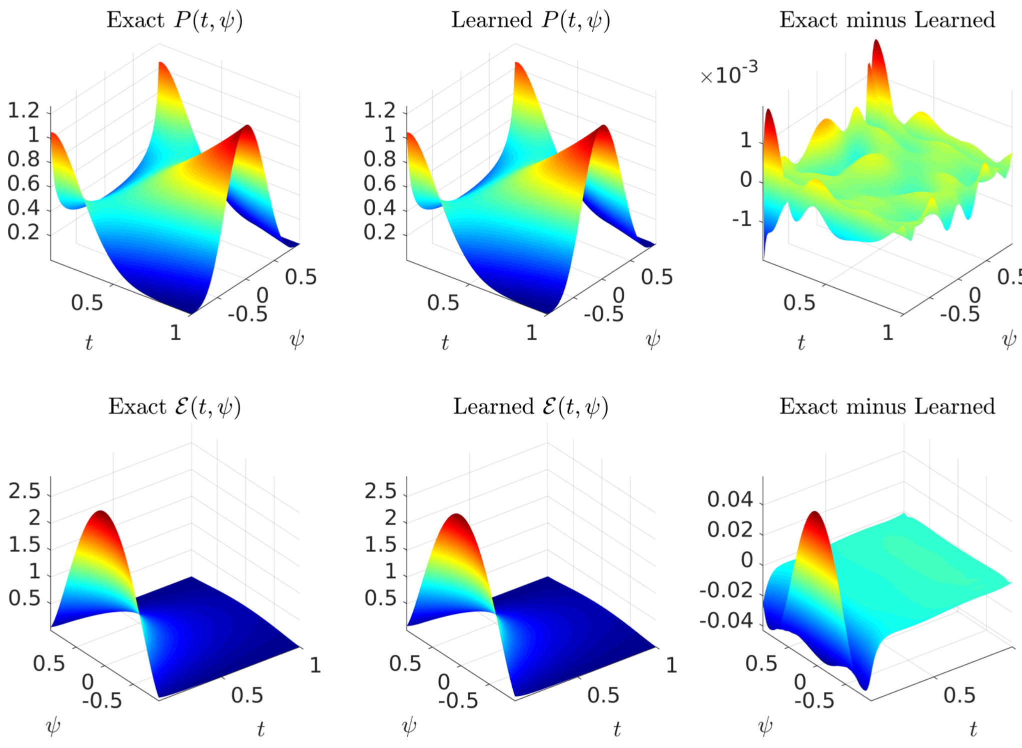


FIG. 5. *Conditional expected dissipation:* The exact probability density function $P(t, \psi)$ alongside the learned one is depicted in the top panels, while the exact and learned conditional expected dissipation $\mathcal{E}(t, \psi)$ are plotted in the bottom panels. It is worth highlighting that the algorithm has seen no data whatsoever on the dissipation coefficient.

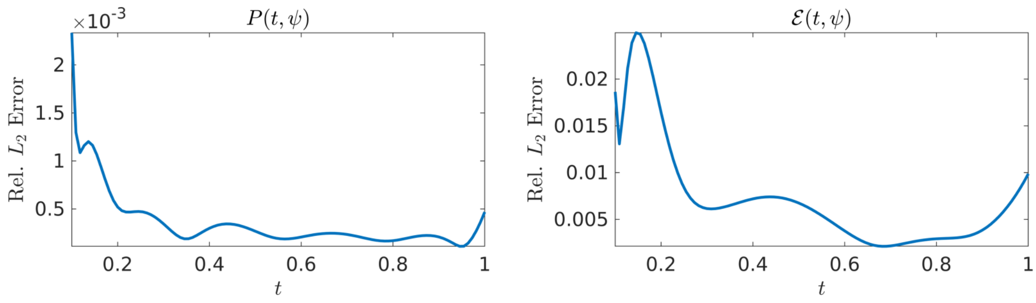


FIG. 6. The relative L_2 error as a function of time for probability density function $P(t, \psi)$ (left) and conditional expected dissipation $\mathcal{E}(t, \psi)$ (right).

As for the training procedure, our experience so far indicates that while training deep neural networks, it is often useful to reduce the learning rate as the training progresses. Specifically, the results reported here are obtained after 10^5, 2×10^5, 3×10^5, and 4×10^5 consecutive epochs of the Adam optimizer [94] with learning rates of 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}, respectively. Each epoch corresponds to one pass through the entire dataset. The total number of iterations of the Adam optimizer is therefore given by 10^6 times the number of data divided by the mini-batch size. The mini-batch size we used is 20 000 and the number of data points is 20 000. Every ten iterations of the optimizer takes around 0.04 on a single NVIDIA Titan X GPU card. The algorithm is capable of reconstructing the probability density function $P(t, \psi)$ as well as the unknown conditional expected diffusion $D(t, \psi)$ with average relative L_2 errors of 1.27×10^{-4} and 1.73×10^{-2}, respectively. The relative L_2 errors in space as a function of time are depicted in Fig. 4. The largest errors are observed near $t = 0$, where the exact PDF is represented by two δ functions, i.e., a singular behavior, and this results in relatively large errors near the initial condition. However, at later times the error decreases significantly. The difficulty of a deep neural network in approximating singular initial conditions is akin to other approximation methods as well, such as finite element methods.

The effects of the number of training points (N) on the accuracy of the learned P and D are shown in Table I and indicate robustness with respect to the number of points. Also, it is to be noted that the method is a regression scheme that regresses the data on PDF while it also regresses the PDF transport equation. As such, a convergence with respect to the number of training points cannot be established. This is further exacerbated by the fact that the neural network parameters are initialized randomly in conjunction with a stochastic gradient descent optimization. Thus, it is virtually impossible to reproduce the results even if the random seed is kept fixed.

We also investigated the effect of noisy training data on the accuracy of learned quantities. To this end, we added an independent centered Gaussian noise to the PDF training data:

$$P(t^n, \psi^n) = P_{AMC}(t^n, \psi^n) + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$ is a zero-mean Gaussian noise with a standard deviation of σ, and P_{AMC} is the PDF according to AMC. The relative L_2 errors for P and $\mathcal{E}$ versus different noise levels are shown in Table II. The computations are carried out for $N = 20\,000$ training points. These results indicate

TABLE III. *Conditional expected dissipation:* Relative L_2 error for P and $\mathcal{E}$ vs number of training points.

Number of Training Points	20000	10000	5000	2500
Relative L_2 error for P	3.89×10^{-4}	3.82×10^{-4}	3.78×10^{-4}	4.55×10^{-4}
Relative L_2 error for $\mathcal{E}$	3.09×10^{-2}	3.23×10^{-2}	2.09×10^{-2}	5.56×10^{-2}

TABLE IV. *Conditional expected dissipation:* Relative L_2 error for P and $\mathcal{E}$ vs levels of noise.

Noise level (σ)	0.1	0.05	0.01
Relative L_2 error for P	2.95×10^{-3}	1.22×10^{-3}	4.43×10^{-4}
Relative L_2 error for $\mathcal{E}$	3.93×10^{-2}	4.41×10^{-2}	1.10×10^{-2}

the robustness of the method with respect to noise in which the conditional expected diffusion can be learned from noisy PDF data.

Figure 5 depicts the exact and the learned conditional expected dissipation $\mathcal{E}(t, \psi)$. It is worth highlighting that the algorithm has seen no data whatsoever on the dissipation coefficient. To obtain the results reported in this figure we are approximating $P(t, \psi)$ and $\mathcal{E}(t, \psi)$ by a deep neural network outputting two variables consisting of ten hidden layers with 100 neurons per each hidden layer (see Fig. 2). As for the activation functions, we use x sigmoid(x). The training procedure is the same as that explained above, while every 10 iterations of the optimizer takes around 0.07. The algorithm is capable of reconstructing the probability density function $P(t, \psi)$ as well as the unknown conditional expected dissipation $\mathcal{E}(t, \psi)$ with average relative L_2 errors of 3.84×10^{-4} and 1.75×10^{-2}, respectively. The relative L_2 errors in space as a function of time are depicted in Fig. 6. Similar to the previous case, the robustness of the learned quantities with respect to the number of training points and to different noise levels are demonstrated in Tables III and IV.

VI. CONCLUDING REMARKS

A data-driven framework is developed for learning unclosed terms for turbulent scalar mixing. In the presented framework the unclosed terms are learned by (i) incorporating the physics, i.e., the PDF transport equation, and (ii) noting some high-fidelity observations on the PDF. We envision that the presented framework can be readily extended to high-dimensional cases involving the mixing of multiple species. Early evidence of this claim can be found in Refs. [19,82], in which the authors circumvent the tyranny of numerical discretization and devise algorithms that are scalable to high dimensions. A similar technique can be applied here while taking advantage of the fact that the data points $\{t^n, \psi^n, P^n\}_{n=1}^N$ lie on a low-dimensional manifold simply because $\psi(t, x)$ is a function from a low-dimensional space [i.e., (t, x)] to the possibly high-dimensional space of species ψ. Moreover, the approach as advocated here is also highly scalable to the big data regimes routinely encountered while studying turbulence simply because the data will be processed in mini-batches. It would be very interesting to implement the methodology as developed here in the context of laboratory experiments on passive scalar mixing, such as those in Refs. [15,95].

All data and codes used in this manuscript will be publicly available on GitHub [96].

ACKNOWLEDGMENTS

The work at Brown University is supported by DARPA EQUiPS Grant No. N66001-15-2-4055 and by AFOSR Grant No. FA9550-17-1-0013. The work at Pitt is sponsored by the American Chemical Society through Petroleum Research Fund Grant No. 59081-DN19 and NSF Grant No. CBET-1609120.

[1] E. E. O'Brien, On the statistical behavior of a dilute reactant in isotropic turbulence, Ph.D. Thesis, The Johns Hopkins University, Baltimore, MD, 1960.
[2] *Turbulence in Mixing Operation*, edited by R. S. Brodkey (Academic Press, New York, 1975).
[3] S. B. Pope, *Turbulent Flows* (Cambridge University Press, Cambridge, UK, 2000).

[4] D. C. Haworth, Progress in probability density function methods for turbulent reacting flows, Prog. Energ. Combust. **36**, 168 (2010).

[5] A. G. Nouri, M. B. Nik, P. Givi, D. Livescu, and S. B. Pope, A self-contained filtered density function, Phys. Rev. Fluids **2**, 094603 (2017).

[6] S. Frolov, V. Frost, and D. Roekaerts, *Micromixing in Turbulent Reactive Flows* (Torus Press, Moscow, 2004).

[7] N. Ansari, F. A. Jaberi, M. R. H. Sheikhi, and P. Givi, Filtered density function as a modern CFD tool, in *Engineering Applications of CFD*, edited by R. S. Maher, Fluid Mechanics and Its Applications Vol. 1 (International Energy and Environment Foundation, Al-Najaf, Iraq, 2011), pp. 1–22.

[8] S. B. Pope, Small scales, many species and the manifold challenges of turbulent combustion, Proc. Combust. Inst. **34**, 1 (2013).

[9] R. S. Miller and J. W. Foster, Survey of turbulent combustion models for large-eddy simulations of propulsive flow fields, AIAA J. **54**, 2930 (2016).

[10] S. S. Girimaji and Y. Zhou, Analysis and modeling of subgrid scalar mixing using numerical data, Phys. Fluids **8**, 1224 (1996).

[11] F. A. Jaberi, R. S. Miller, C. K. Madnia, and P. Givi, Non-Gaussian scalar statistics in homogeneous turbulence, J. Fluid Mech. **313**, 241 (1996).

[12] S. L. Christie and J. A. Domaradzki, Scale dependence of the statistical character of turbulent fluctuations in thermal convection, Phys. Fluids **6**, 1848 (1994).

[13] A. Eckstein and P. P. Vlachos, Digital particle image velocimetry (DPIV) robust phase correlation, Meas. Sci. Technol. **20**, 055401 (2009).

[14] F. Pereira, M. Gharib, D. Dabiri, and D. Modarress, Defocusing digital particle image velocimetry: A 3-component 3-dimensional DPIV measurement technique. Application to bubbly flows, Exp. Fluids **29**, S078 (2000).

[15] C. Tong, Measurements of conserved scalar filtered density function in a turbulent jet, Phys. Fluids **13**, 2923 (2001).

[16] D. C. Psichogios and L. H. Ungar, A hybrid neural network–first principles approach to process modeling, AIChE J. **38**, 1499 (1992).

[17] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. **9**, 987 (1998).

[18] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. **375**, 1339 (2018).

[19] E. Weinan, J. Han, and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. **5**, 349 (2017).

[20] Z. Long, Y. Lu, X. Ma, and B. Dong, PDE-Net: Learning PDEs from data, arXiv:1710.09668.

[21] M. Baymani, A. Kerayechian, and S. Effati, Artificial neural networks approach for solving Stokes problem, Appl. Math. **1**, 288 (2010).

[22] M. Chiaramonte and M. Kiener, Solving differential equations using neural networks, Machine Learning Project (2018), http://cs229.stanford.edu/proj2013/ChiaramonteKiener-SolvingDifferentialEquationsUsingNeuralNetworks.pdf.

[23] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. **3**, e1602614 (2017).

[24] S. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz, Data-driven identification of parametric partial differential equations, SIAM J. Appl. Dyn. Syst. **18**, 643 (2019).

[25] S. Pan and K. Duraisamy, Data-driven discovery of closure models, SIAM J. Appl. Dyn. Syst. **17**, 2381 (2018).

[26] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. **378**, 686 (2019).

[27] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. **19**, 932 (2018).

[28] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, J. Comput. Phys. **335**, 736 (2017).

[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, J. Comput. Phys. **348**, 683 (2017).

[30] H. Owhadi, Bayesian numerical homogenization, Multiscale Model. Simul. **13**, 812 (2015).

[31] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, 2006), Vol. 1.

[32] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Numerical Gaussian processes for time-dependent and nonlinear partial differential equations, SIAM J. Sci. Comput. **40**, A172 (2018).

[33] M. Raissi and G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, J. Comput. Phys. **357**, 125 (2018).

[34] H. Owhadi, C. Scovel, and T. Sullivan, Brittleness of Bayesian inference under finite information in a continuous world, Electron. J. Stat. **9**, 1 (2015).

[35] G. Pang, L. Yang, and G. E. Karniadakis, Neural-net-induced Gaussian process regression for function approximation and PDE solution, J. Comput. Phys. **384**, 270 (2019).

[36] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations, arXiv:1711.10566.

[37] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations, arXiv:1711.10561.

[38] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, arXiv:1801.06637.

[39] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks **2**, 359 (1989).

[40] Y. Zhu and N. Zabaras, Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. **366**, 415 (2018).

[41] T. Hagge, P. Stinis, E. Yeung, and A. M. Tartakovsky, Solving differential equations with unknown constitutive relations as recurrent neural networks, arXiv:1710.02242.

[42] R. Tripathy and I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, J. Comput. Phys. **375**, 565 (2018).

[43] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, Data-driven forecasting of high-dimensional chaotic systems with long-short term memory networks, Proc. Math Phys. Eng. Sci. **474**, 2213 (2018).

[44] E. J. Parish and K. Duraisamy, A paradigm for data-driven predictive modeling using field inversion and machine learning, J. Comput. Phys. **305**, 758 (2016).

[45] K. Duraisamy, Z. J. Zhang, and A. P. Singh, New approaches in turbulence and transition modeling using data-driven techniques, in *53rd AIAA Aerospace Sciences Meeting* (AIAA, Reston, VA, 2015), p. 1284.

[46] J. Ling, A. Kurzawski, and J. Templeton, Reynolds averaged turbulence modeling using deep neural networks with embedded invariance, J. Fluid Mech. **807**, 155 (2016).

[47] Z. J. Zhang and K. Duraisamy, Machine learning methods for data-driven turbulence modeling, in *22nd AIAA Computational Fluid Dynamics Conference* (AIAA, Reston, VA, 2015), p. 2460.

[48] M. Milano and P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. **182**, 1 (2002).

[49] P. Perdikaris, D. Venturi, and G. E. Karniadakis, Multifidelity information fusion algorithms for high-dimensional systems and massive data sets, SIAM J. Sci. Comput. **38**, B521 (2016).

[50] R. Rico-Martinez, J. Anderson, and I. Kevrekidis, Continuous-time nonlinear signal processing: A neural network based approach for gray box identification, in *Neural Networks for Signal Processing 1994, IV. Proceedings of the 1994 IEEE Workshop* (IEEE, New York, 1994), pp. 596–605.

[51] J. Ling and J. Templeton, Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty, Phys. Fluids **27**, 085103 (2015).

[52] J.-X. Wang, J. Wu, J. Ling, G. Iaccarino, and H. Xiao, A comprehensive physics-informed machine learning framework for predictive turbulence modeling, Phys. Rev. Fluids **3**, 074602 (2018).

[53] R. S. Beidokhti and A. Malek, Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques, J. Franklin Inst. **346**, 898 (2009).
[54] R. Kondor, N-body networks: A covariant hierarchical neural network architecture for learning atomic potentials, arXiv:1803.01588.
[55] R. Kondor and S. Trivedi, On the generalization of equivariance and convolution in neural networks to the action of compact groups, arXiv:1802.03690.
[56] M. Hirn, S. Mallat, and N. Poilvert, Wavelet scattering regression of quantum chemical energies, Multiscale Model. Simul. **15**, 827 (2017).
[57] S. Mallat, Understanding deep convolutional networks, Phil. Trans. R. Soc. A **374**, 20150203 (2016).
[58] C. Dopazo, Non-isothermal turbulent reactive flows: Stochastic approaches, Ph.D. Thesis, Department of Mechanical Engineering, State University of New York at Stony Brook, 1973.
[59] J. Janicka, W. Kolbe, and W. Kollmann, Closure of the transport equation for the probability density function of turbulent scalar fields, J. Non-Equilib. Thermodyn. **4**, 47 (1979).
[60] S. B. Pope, An improved turbulent mixing model, Combust. Sci. Technol. **28**, 131 (1982).
[61] G. Kosály and P. Givi, Modeling of turbulent molecular mixing, Combust. Flame **70**, 101 (1987).
[62] P. Givi and P. A. McMurtry, Non-premixed reaction in homogeneous turbulence: Direct numerical simulations, AIChE J. **34**, 1039 (1988).
[63] A. T. Norris and S. B. Pope, Turbulent mixing model based on ordered pairing, Combust. Flame **83**, 27 (1991).
[64] S. S. Girimaji, On the modeling of scalar diffusion in isotropic turbulence, Phys. Fluids A **4**, 2529 (1992).
[65] S. S. Girimaji, A mapping closure for turbulent scalar mixing using a time-evolving reference field, Phys. Fluids A **4**, 2875 (1992).
[66] F. A. Jaberi and P. Givi, Inter-layer diffusion model of scalar mixing in homogeneous turbulence, Combust. Sci. Technol. **104**, 249 (1995).
[67] S. Subramaniam and S. B. Pope, A mixing model for turbulent reactive flows based on Euclidean minimum spanning trees, Combust. Flame **115**, 487 (1998).
[68] S. B. Pope, A model for turbulent mixing based on shadow-position conditioning, Phys. Fluids **25**, 110803 (2013).
[69] S. Tavoularis and S. Corrsin, Experiments in nearly homogenous turbulent shear flow with a uniform mean temperature gradient. Part 1, J. Fluid Mech. **104**, 311 (1981).
[70] V. Eswaran and S. B. Pope, Direct numerical simulations of the turbulent mixing of a passive scalar, Phys. Fluids **31**, 506 (1988).
[71] P. A. McMurtry and P. Givi, Direct numerical simulations of mixing and reaction in a nonpremixed homogeneous turbulent flow, Combust. Flame **77**, 171 (1989).
[72] S. L. Christie and J. A. Domaradzki, Numerical evidence for the nonuniversality of the soft/hard turbulence classification for thermal convection, Phys. Fluids A **5**, 412 (1993).
[73] T. H. Solomon and J. P. Gollub, Thermal boundary layers and heat flux in turbulent convection: The role of recirculating flows, Phys. Rev. A **43**, 6683 (1991).
[74] S. T. Thoroddsen and C. W. Van Atta, Exponential tails and skewness of density-gradient probability density functions in stably stratified turbulence, J. Fluid Mech. **244**, 547 (1992).
[75] Jayesh and Z. Warhaft, Probability Distribution of a Passive Scalar in Grid-Generated Turbulence, Phys. Rev. Lett. **67**, 3503 (1991).
[76] Jayesh and Z. Warhaft, Probability distribution, conditional dissipation, and transport of passive temperature fluctuations in grid-generated turbulence, Phys. Fluids A **4**, 2292 (1992).
[77] M. Raissi, H. Babaee, and G. E. Karniadakis, Parametric Gaussian process regression for big data, Comput. Mech. **64**, 409 (2019).
[78] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. R. Soc. A **473**, 20160751 (2017).
[79] M. Raissi and G. Karniadakis, Deep multi-fidelity Gaussian processes, arXiv:1604.07484.
[80] M. Gulian, M. Raissi, P. Perdikaris, and G. Karniadakis, Machine learning of space-fractional differential equations, arXiv:1808.00931.

[81] M. Raissi, A. Yazdani, and G. E. Karniadakis, Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data, arXiv:1808.04327.
[82] M. Raissi, Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations, arXiv:1804.07010.
[83] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, arXiv:1801.01236.
[84] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, Deep learning of vortex-induced vibrations, J. Fluid Mech. **861**, 119 (2019).
[85] A. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 5595 (2017).
[86] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467.
[87] R. H. Kraichnan, Closures for probability distributions, Bull. Am. Phys. Soc. **34**, 2298 (1989).
[88] H. Chen, S. Chen, and R. H. Kraichnan, Probability Distribution of a Stochastically Advected Scalar Field, Phys. Rev. Lett. **63**, 2657 (1989).
[89] S. B. Pope, Mapping closures for turbulent mixing and reaction, Theor. Comp. Fluid Dyn. **2**, 255 (1991).
[90] R. S. Miller, S. H. Frankel, C. K. Madnia, and P. Givi, Johnson-Edgeworth translation for probability modeling of binary scalar mixing in turbulent flows, Combust. Sci. Technol. **91**, 21 (1993).
[91] T.-L. Jiang, F. Gao, and P. Givi, Binary and trinary scalar mixing by Fickian diffusion—Some mapping closure results, Phys. Fluids A **4**, 1028 (1992).
[92] P. Ramachandran, B. Zoph, and Q. V. Le, Searching for activation functions, arXiv:1710.05941.
[93] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, On the expressive power of deep neural networks, in *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of Proceedings of Machine Learning Research (International Convention Centre, Sydney, Australia, 2017), pp. 2847–2854.
[94] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
[95] A. G. Rajagopalan and C. Tong, Experimental investigation of scalar-scalar-dissipation filtered joint density function and its transport equation, Phys. Fluids **15**, 227 (2003).
[96] https://github.com/maziarraissi/DeepTurbulence.