Physica D 443 (2023) 133570

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Output-weighted and relative entropy loss functions for deep learning precursors of extreme events

Samuel H. Rudy *, Themistoklis P. Sapsis

Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States of America

ARTICLE INFO

Article history:
Received 6 May 2022
Received in revised form 8 August 2022
Accepted 19 October 2022
Available online 2 November 2022
Communicated by B. Hamzi

Keywords:
Computational methods
Machine learning
Extreme events

ABSTRACT

Many scientific and engineering problems require accurate models of dynamical systems with rare and extreme events. Such problems present a challenging task for data-driven modeling, with many naive machine learning methods failing to predict or accurately quantify such events. One cause for this difficulty is that systems with extreme events, by definition, yield imbalanced datasets and that standard loss functions easily ignore rare events. That is, metrics for goodness of fit used to train models are not designed to ensure accuracy on rare events. This work seeks to improve the performance of regression models for extreme events by considering loss functions designed to highlight outliers. We propose a novel loss function, the adjusted output weighted loss, and extend the applicability of relative entropy based loss functions to systems with low dimensional output. The proposed functions are tested using several cases of dynamical systems exhibiting extreme events and shown to significantly improve accuracy in predictions of extreme events.

1. Introduction

Accurate prediction and quantification of extreme events are critical tasks in many areas of science and engineering. Specific cases include the study of extreme weather patterns [1], turbulence [2], macroeconomic fluctuations [3], rouge waves [4], and many others [5,6]. Recent research has developed tools for experimental design aimed towards uncertainty quantification in tail regions of systems with extreme events [7–9], prediction in the sense of classification of upcoming events in turbulent fluid flows [10–12], and regression problems for systems exhibiting extreme events [13,14].

These recent works fall into the broader category of data-driven approaches to dynamical systems and fluid dynamics [15,16]. Uses for such methods are motivated by cases in which physics based models fail due to intractable complexity, computational requirements, or insufficient measurements. In these cases machine learning and in particular deep learning offer a potential tool for improved predictive modeling. Successful applications of deep learning to problems in dynamical systems include flow reconstruction [17], physics informed neural networks [18], closure models [19,20], sub-grid scale models [21,22], climate modeling [23], operator inference [24], and embedding and lifting transformations [25–27].

Several recent works have explored the use of specific loss functions for training prediction models in the context of extreme events. Guth and Sapsis [12] develop the maximum adjusted area under the precision recall curve and show that it is effective in predicting extreme events in systems including the Kolmogorov flow and Majda–McLaughlin–Tabak model [28]. However, their proposed metric is not differentiable and not well approximated by small samples. Implementation for high dimensional models such as neural networks would therefore be challenging. Doan et al. [13] use a physics informed loss function to improve the accuracy of echo-state networks for forecasting a Galerkin model of a turbulent flow with intermittent quasi-laminar states. While effective, this approach is constrained to problems where there is a known dynamic model for the quantity being predicted. Authors of [14] use a relative entropy based loss function to forecast the truncated Korteweg–de Vries equation, a simplified model of turbulent surface waves. This is shown to significantly improve performance, but requires a high dimensional target quantity. More recently, the use of various model architectures for predicting extreme events has been studied in [29].

In this work we seek to develop more broadly applicable loss functions and evaluate their performance on several challenging test problems. While the loss functions proposed in this work may be applied to arbitrary regression models, the included examples problems both employ neural networks. We assume the reader is familiar with common deep learning techniques including recurrent neural networks, stochastic optimization, and early stopping. The unfamiliar reader may find an excellent and free online reference in [30]. In particular, we make use of long-short-term-memory networks [31] for each of the test cases used

* Corresponding author.
E-mail address: shrudy@mit.edu (S.H. Rudy).

https://doi.org/10.1016/j.physd.2022.133570
0167-2789/

in this work. The results could almost certainly be improved via more carefully thought out network structures, training, and other user decisions [29]. However, such considerations are not the focus of this work which focuses solely on the effect of loss functions.

The paper is organized as follows; In Section 2 we outline the motivation for extreme event specific loss functions and develop methods including output weighted variations of the mean square error and a relative entropy method based on work in [14]. We also discuss error metrics for evaluating the proposed loss functions. In Section 3 we present results of the proposed loss functions applied to two test cases; Kolmogorov flow at Reynolds number $Re = 40$ and the flow around a square cylinder at $Re = 5000$. A discussion of the results, limitations, and outlook is presented in Section 4.

2. Methods

In this section we outline the proposed loss functions used for training neural networks to predict and quantify extreme events. These include two weighted variations of the mean square error as well as a relative entropy, also known as KL-divergence, based loss. We also describe methods for approximating the density function of the target variable y and the metrics we use to measure accuracy of the trained networks.

2.1. Problem statement

Let (X, Σ_x, μ) be a probability space with $X \subseteq \mathbb{R}^n$ and μ absolutely continuous with probability density function p_x. For some unknown function f, we have a dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ where $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$, $\mathbf{x}_i \sim p_x$, $y = \{y_1, \dots, y_m\}$ with $y_i = f(\mathbf{x}_i)$, perhaps perturbed by measurement noise. We will make use of the fact that since $p(\mathcal{D}) = \prod_i p_x(\mathbf{x}_i)$ we have $\mathbb{E}_{\mathcal{D}}[\frac{1}{m}\sum g(\mathbf{x}_i)] = \mathbb{E}_{\mathbf{x}}[g(\mathbf{x})]$ for any g such that the expectation is finite. That is, empirical averages over $\mathcal{D}$ are unbiased approximations of integrals over p_x.

We are interested in parametric models $\hat{f}$ approximating f which accurately predict and quantify outlier values of y and which accurately capture the induced probability density function through a measure transformation: $p_y(y) = d/dy\, \mu(f^{-1}(-\infty, y))$. To this end, both the structure of the model and training parameters have important effects. We seek to develop objective functions tailored for extreme event prediction that accurately capture extreme events, are differentiable, and may be approximated from finite datasets.

Throughout the remainder of the paper, we will use the terms true and false positive and negative to describe various results from a continuous regression model. In this context, we loosely define a true positive to be a prediction $\hat{f}(\mathbf{x})$ such that $p_y(f(\mathbf{x})) \sim p_y(\hat{f}(\mathbf{x})) \ll 1$. That is, when both the true and predicted values of y are rare. Likewise, a true negative is when $p_y(f(\mathbf{x})) \sim p_y(\hat{f}(\mathbf{x})) \sim \mathcal{O}(1)$, a false positive is when $p_y(f(\mathbf{x})) \sim \mathcal{O}(1)$ and $p_y(\hat{f}(\mathbf{x})) \ll 1$, and a false negative is when $p_y(f(\mathbf{x})) \ll 1$ and $p_y(\hat{f}(\mathbf{x})) \sim \mathcal{O}(1)$. In some places, these definitions will be made rigorous by applying thresholds to $p_y(f(\mathbf{x}))$ and $p_y(\hat{f}(\mathbf{x}))$.

2.2. Output-weighted variations of mean square error

The mean square error is the most common loss function used for training regression models with real valued outputs. It is given by,

$$L_{MSE}(\hat{f}) = \mathbb{E}_{\mathbf{x}}\left[e_{\hat{f}}(\mathbf{x})^2\right] = \int_X e_{\hat{f}}(\mathbf{x})^2 p_x(\mathbf{x})\, d\mathbf{x} = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{m}\sum_{\mathcal{D}} e_{\hat{f}}(\mathbf{x}_i)^2\right], \quad (1)$$

where $e_{\hat{f}} = (f(\mathbf{x}) - \hat{f}(\mathbf{x}))$. Squaring the error makes L_{MSE} more sensitive to true outliers than the mean absolute error, but if outlier values make up a small fraction of the total dataset then L_{MSE} may still be small while missing large y. Moreover, if some rare values of y are not separated from the core of p_y by substantial distance then they may be missed with little added error. To better see this, consider the case where μ admits a disintegration over the induced measure on y. Then we can express L_{MSE} as an integral over Y of the regular conditional expectation of the square error. That is,

$$\begin{aligned} L_{MSE}(\hat{f}) &= \mathbb{E}_y\left[\mathbb{E}_{\mathbf{x}}\left[e_{\hat{f}}(\mathbf{x})^2 \middle| f(\mathbf{x}) = y\right]\right] \\ &= \int_Y p_y(y)\, \mathbb{E}_{\mathbf{x}}\left[e_{\hat{f}}(\mathbf{x})^2 \middle| f(\mathbf{x}) = y\right] dy, \end{aligned} \quad (2)$$

where the conditional expectation is defined using the disintegration of p_x over p_y [32]. For rare events, the value of $p_y(y)$ is small, allowing large error in the prediction of such events without significantly affecting L_{MSE}.

The insensitivity of L_{MSE} to rare events may be mitigated via introducing a weighting function. Specifically, consider the case where the square error is weighted according to $p_y(f(\mathbf{x}))^{-1}$. The resulting function is given by,

$$\begin{aligned} L_{OW}(\hat{f}) &= \mathbb{E}_{\mathbf{x}}\left[p_y(f(\mathbf{x}))^{-1} e_{\hat{f}}(\mathbf{x})^2\right] \\ &= \int_Y p_y(y)\, \mathbb{E}_{\mathbf{x}}\left[p_y(f(\mathbf{x}))^{-1} e_{\hat{f}}(\mathbf{x})^2 \middle| f(\mathbf{x}) = y\right] dy \\ &= \int_Y \mathbb{E}_{\mathbf{x}}\left[e_{\hat{f}}(\mathbf{x})^2 \middle| f(\mathbf{x}) = y\right] dy. \end{aligned} \quad (3)$$

Note that the expression $p_y(f(\mathbf{x}))|_{f(\mathbf{x})=y}$ is simply $p_y(y)$ and therefore cancels the $p_y(y)$ term outside the conditional expectation. We call the expression given by Eq. (3) the output-weighted loss, L_{OW} since the square error is weighed by the inverse of the likelihood of the true output $f(\mathbf{x})$. Expressions with similar form have been used for sequential sampling strategies for rare events [8,9]. However, these works were focused on experimental design and used least squares or Gaussian process regression to model f. Eq. (3) is also related to oversampling techniques commonly used for classification problems with imbalanced data [33].

As a cost function, Eq. (3) has some potential drawbacks. In particular, the weight given to each error is proportional only to the inverse likelihood of the true value, $p_y(f(\mathbf{x}))$, and is independent from $p_y(\hat{f}(\mathbf{x}))$. Thus, error accumulated on false negatives is penalized far more than that made on false positives. As we will show in Section 3, minimizing Eq. (3) often yields models that over-predict rare events.

The problem of false positives may be addressed with the inclusion of a second term weighing the square error. The weight function $1/p_y(f(\mathbf{x}))$ in Eq. (3) amplifies any error realized on examples with rare y. This included true positive and false negative predictions. To distinguish between true negative and false positive predictions, we require a weight function that depends on $p_y(\hat{f}(\mathbf{x}))$. The ratio $p_y(f(\mathbf{x}))/p_y(\hat{f}(\mathbf{x}))$ is large only in the case where the likelihood of the predicted output is lower than that of the true output. Thus, the expression given by,

$$L_{FP}(\hat{f}) = \mathbb{E}_{\mathbf{x}}\left[\frac{p_y(f(\mathbf{x}))}{p_y(\hat{f}(\mathbf{x}))} e_{\hat{f}}(\mathbf{x})^2\right], \quad (4)$$

is a measure of the error made on false positive predictions. Summing Eq. (3) with Eq. (4) gives,

$$\begin{aligned} L_{AOW}(\hat{f}) &= L_{OW}(\hat{f}) + L_{FP}(\hat{f}) \\ &= \mathbb{E}_{\mathbf{x}}\left[\left(\frac{1}{p_y(f(\mathbf{x}))} + \frac{p_y(f(\mathbf{x}))}{p_y(\hat{f}(\mathbf{x}))}\right) e_{\hat{f}}(\mathbf{x})^2\right] \end{aligned} \quad (5)$$

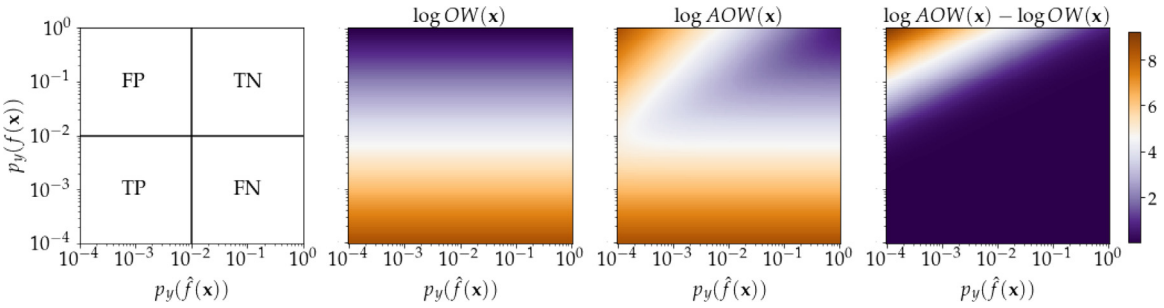

Fig. 1. Visualization of weights used by L_{OW} and L_{AOW}. Left most figure shows hypothetical boundaries for true/false positive/negative regions. Pseudocolor plots show output weight, adjusted output weight, and difference. Disagreement is primarily within the false positive region.

which we will call the *adjusted output weighted loss*, or L_{AOW}. For true positives, when both $p_y(f(\mathbf{x}))$ and $p_y(\hat{f}(\mathbf{x}))$ are small, the second term in the parentheses is $\mathcal{O}(1) \ll p_y(f(\mathbf{x}))^{-1}$. The integrand therefore agrees with that of Eq. (3). For true negatives and false negatives, the integrand is also of similar magnitude to that in Eq. (3). However, for false positives, the second term in the parenthesis is large. The expression therefore penalizes errors made as false positive predictions to a greater degree than Eq. (3).

The added penalization of false positives comes at the cost of increased complexity. Eq. (3) is a weighted least squares problem and does not add any computational complexity to Eq. (1) beyond pre-computing $\{p_y(y_i)\}_{i=1}^m$. In particular, Eq. (3) has closed form solution for linear problems. This is not the case for Eq. (5), where iterative optimization is required. For $\hat{f}$ depending nonlinearly on parameters, such as neural networks, the difference is less important, as iterative methods must be used in either case.

Differences between the weighting functions in the integrands of Eq. (3) and (5) are visualized in Fig. 1. We plot values of the each term using hypothetical values of $p_y(f(\mathbf{x}))$ and $p_y(\hat{f}(\mathbf{x}))$. The left most plot illustrates a rough partitioning of the domain into true/false positives and negatives. Note that many expressions of the likelihoods of the true and predicted y could be constructed to find novel loss functions. The authors suspect that this would be a fruitful research direction, but it is not within the scope of this work.

2.3. Approximating p_y

Computation of the two weighted variations of the mean square error in the previous section requires evaluating p_y at points y_i in the training set. For Eq. (3), this may be done offline using any off-the-shelf density estimation technique. Evaluating (5) further requires $p_y \circ \hat{f}$, which changes during training and must be evaluated rapidly. Gradient based optimization also requires evaluating $p_y' \circ \hat{f}$. Thus, minimizing Eq. (5) requires a low-computational-cost differentiable approximation of p_y. This rules out non-parametric density estimates such as kernel density estimation (due to slowness), histograms (due to lack of differentiability), and k-nearest-neighbors (slow and not differentiable). We instead use a combined method of nonparametric estimation of $log(p_y(y_i))$ which we then fit using a Gaussian process with few collocation points, allowing for rapid evaluation of p_y and its derivative.

Estimates of $p_y(y_i)$ are initially made using a histogram of $\{y_i\}_{i=1}^m$ with $n_b = 100$ evenly sized bins. Defining $B_i = [b_i, b_{i+1})$ for $i = 1, \ldots, 99$ and $B_{100} = [b_{100}, b_{101}]$ where b_i are evenly spaced values between $min(y_i)$ and $max(y_i)$ we have for $y \in B_i$,

$$\log \hat{p}(c_i) \approx \log\left(\frac{\left|\{j : y_j \in B_i\}\right|}{ml}\right) \tag{6}$$

where $l = b_{i+1} - b_i$ and $c_i = (b_i + b_{i+1})/2$.

Following estimation of the log densities via histogram, the values at the center of each bin are fit to a Gaussian process [34]. We use the Matern-52 kernel with additional white heteroscedastic noise. The mean function of the Gaussian process is set to be the log of machine epsilon, enforcing that the approximation goes to zero away from the sampled data. That is,

$$\log \hat{p}_y(y) \sim \mathcal{GP}\left(\mu, k(y, y') + \sigma_w^2(y)\delta_{y,y'}\right), \tag{7}$$

where $\mu = \log(10^{-16})$, $\sigma_w(y)$ is the white noise term and $\delta_{y,y'}$ is the Kronecker delta function. Fitting the logarithm of the probability rather than the probability helps to get accurate estimates in the tails of p_y. We obtain estimates of the density at sample points y using the conditional mean of the Gaussian process;

$$\begin{aligned}\hat{p}_y(y) &= \exp\left(k(y, \mathbf{c})\left(k(\mathbf{c}, \mathbf{c}) + \sigma_w^2(\mathbf{c})\mathbf{I}\right)^{-1} \log \hat{p}_y(\mathbf{c}) + \mu\right) \\ &= \exp\left(k(y, \mathbf{c})\boldsymbol{\alpha} + \log(10^{-16})\right).\end{aligned} \tag{8}$$

where $c_i = (b_i + b_{i+1})/2$ and vector $\boldsymbol{\alpha}$ is pre-computed and stored. The expected gradient of the exponent is given by simply differentiating the kernel [35]. This allows for queries $\hat{p}_y(y)$ and its gradient with the simple computation of $k(y, \mathbf{c})$ and $k'(y, \mathbf{c})$. In practice we found training to be more stable when a floor was set for the value of $p_y(y)$. Examples in this work all used an effective density equal to $\hat{p}_y(y) + 10^{-5}$ which has negligible effect of most events.

We note that the same process could be easily implemented for y with dimension greater than one but still low. Initial estimates of p_y at training points of the Gaussian process could be taken with any standard non-parametric density estimation [36] and subsequently fit to a GP. However, for higher dimensional y, both density approximation and Gaussian process interpolation become non-trivial. It is possible that in these cases the density p_y could be substituted for that of a relevant observable $g(y)$, but such work is beyond the scope of the present manuscript.

2.4. A relative entropy based loss function

The use of relative entropy as a loss function for neural networks was explored in [14]. Qi and Majda used the relative entropy (i.e. KL-divergence) between truth and prediction, after applying the soft-max function. Specifically, for $y \in \mathbb{R}^s$, [14] uses a loss function defined for a single datapoint by,

$$\begin{aligned}L_{QM}(\mathbf{x}) &= KL\left(\sigma(f(\mathbf{x})) \parallel \sigma(\hat{f}(\mathbf{x}))\right) + \alpha KL\left(\sigma(-f(\mathbf{x})) \parallel \sigma(-\hat{f}(\mathbf{x}))\right) \\ &= \sum\left(\sigma(f(\mathbf{x}))\log\left(\frac{\sigma(f(\mathbf{x}))}{\sigma(\hat{f}(\mathbf{x}))}\right)\right. \\ &\quad \left.+\alpha\sigma(-f(\mathbf{x}))\log\left(\frac{\sigma(-f(\mathbf{x}))}{\sigma(-\hat{f}(\mathbf{x}))}\right)\right)\end{aligned} \tag{9}$$

where log is taken elementwise, the sum is over s dimensions of the vector enclosed in the parenthesis, and the soft-max function, σ, is defined by,

$$\sigma(y) = \frac{exp(y)}{\sum exp(y)} \tag{10}$$

The use of σ weights outputs by the exponent of their magnitude, thus ensuring the loss focuses on accurate learning of large magnitude features. This approach is extremely effective in [14], where y the solution to a PDE and thus high dimensional. However, it may not be applied directly in the case of scalar output y. This is because for any $y \in \mathbb{R}^1$, $\sigma(y) = 1$. Moreover, Eq. (9) is only able to weight extreme values within one output sample, rather than comparing multiple y. However, it is possible to derive similar loss functions where a soft-max like operator is applied to multiple samples, rather than the indices within a single sample.

Let us assume that f and $\hat{f}$ are such that $\mathbb{E}_x[exp(f(\mathbf{x}))]$ and $\mathbb{E}_x[exp(\hat{f}(\mathbf{x}))]$ are finite and non-zero. Note that this is a mild assumption that holds on a set containing $L^\infty(X)$. Then we can define an operator $G(f)$ by,

$$\begin{aligned} G(f)(\mathbf{x}) &= e^{f(\mathbf{x})} p_x(\mathbf{x}) \left(\int_X e^{f(\mathbf{x})} p_x(\mathbf{x})\, d\mathbf{x} \right)^{-1} \\ &= e^{f(\mathbf{x})} p_x(\mathbf{x})\, \mathbb{E}_x\left[e^{f(\mathbf{x})}\right]^{-1}. \end{aligned} \tag{11}$$

This operator acts as a continuous analog of the soft-max function. Functions in the range of G are probability density functions on X where the value of $G(f)(\mathbf{x})$ is proportional to the sample density $p_x(\mathbf{x})$ and the exponent of $f(\mathbf{x})$. $G(f)$ therefore has mass concentrated on those values of $\mathbf{x}$ whose likelihood under p_x is not vanishing and whose image under f is large, or in other words, extreme. Note that in the case where f is linear the operator defined in Eq. (11) is known as exponential tilting [37], which has previously been used in importance sampling [38]. We define the relative entropy loss as,

$$L_{RE}(\hat{f}) = KL\left(G(f) \parallel G(\hat{f})\right). \tag{12}$$

Note that compared to Eq. (9) used in [14], the normalization of $G(f)$ used in Eq. (12) is taken across the input space X rather than dimensions of y. This allows for exponential weighting of outputs by their magnitude even in the case of scalar y. We are interested in minimizing Eq. (12) with respect to $\hat{f}$. Expanding Eq. (12) and ignoring terms whose value does not depend on $\hat{f}$ we find,

$$\begin{aligned} L_{RE}(\hat{f}) &= \mathbb{E}_{\mathbf{x}}\left[\frac{e^{f(\mathbf{x})}}{\mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})}\right]} \log\left(\frac{e^{f(\mathbf{x})} p_x(\mathbf{x})\, \mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right]}{e^{\hat{f}(\mathbf{x})} p_x(\mathbf{x})\, \mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})}\right]} \right) \right] \\ &\propto \mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})} \log\left(e^{f(\mathbf{x}) - \hat{f}(\mathbf{x})}\, \mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] \right) \right] \\ &= \mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})}(f(\mathbf{x}) - \hat{f}(\mathbf{x})) + e^{f(\mathbf{x})} \log\left(\mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] \right) \right]. \end{aligned} \tag{13}$$

Individual expectations in the above expression may be estimated with sums over the dataset. However, the term inside the log is problematic. By Jensen's inequality,

$$\begin{aligned} \log\left(\mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] \right) &= \log\left(\mathbb{E}_{\mathcal{D}}\left[\frac{1}{m} \sum_{i=1}^{m} e^{\hat{f}(\mathbf{x}_i)} \right] \right) \\ &\geq \mathbb{E}_{\mathcal{D}}\left[\log\left(\frac{1}{m} \sum_{i=1}^{m} e^{\hat{f}(\mathbf{x}_i)} \right) \right] \end{aligned} \tag{14}$$

where the second and third expectations are over the random samples $\mathcal{D}$. Thus, the expected log of the empirical average of $exp(\hat{f}(\mathbf{x}))$ is an underestimate. We therefore find an upper bound for Eq. (14) that may be accurately approximated and minimized. Note that for any α, log is bounded above by its first order Taylor expansion about α. Therefore,

$$\begin{aligned} \log\left(\mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] \right) &= \log\left(\alpha + \left(\mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] - \alpha \right) \right) \\ &\leq \log(\alpha) + \frac{\mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] - \alpha}{\alpha}. \end{aligned} \tag{15}$$

The error in the Taylor expansion and thus tightness of the bound is on the order of $(\mathbb{E}_{\mathbf{x}} \exp(\hat{f}(\mathbf{x})) - \alpha)/\alpha^2$. We therefore want to pick some α as close to $\mathbb{E}_{\mathbf{x}} \exp(\hat{f}(\mathbf{x}))$ as possible. Consider $\alpha = \mathbb{E}_{\mathbf{x}} \exp(f(\mathbf{x}))$ which for $f \approx \hat{f}$ we assume will be close. In this case the upper bound for L_{RE} simplifies dramatically to,

$$\begin{aligned} L_{RE}(\hat{f}) &\leq \mathbb{E}_{\mathbf{x}}\Bigg[e^{f(\mathbf{x})}(f(\mathbf{x}) - \hat{f}(\mathbf{x})) \\ &\quad + e^{f(\mathbf{x})} \left(\log\left(\mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})}\right] \right) + \frac{\mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})}\right] - \mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})}\right]}{\mathbb{E}_{\mathbf{x}}\left[e^{f(\mathbf{x})}\right]} \right) \Bigg] \\ &= \mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})} - e^{f(\mathbf{x})} \hat{f}(\mathbf{x}) \right], \end{aligned} \tag{16}$$

where we have ignored terms that do not depend on $\hat{f}$. In particular, the log expectation term, $\log(\alpha)$, has been removed since it does not depend on parameters. The remaining terms are easily approximated from dataset $\mathcal{D}$ by,

$$\begin{aligned} L_{RE}(\hat{f}) &= \mathbb{E}_{\mathbf{x}}\left[e^{\hat{f}(\mathbf{x})} - e^{f(\mathbf{x})} \hat{f}(\mathbf{x}) \right] \\ &= \mathbb{E}_{\mathcal{D}}\left[\frac{1}{m} \sum_{\mathcal{D}} \left(e^{\hat{f}(\mathbf{x}_i)} - e^{f(\mathbf{x}_i)} \hat{f}(\mathbf{x}_i) \right) \right]. \end{aligned} \tag{17}$$

Following [14] we note that the relative entropy loss only focuses the error on large positive values of $f(\mathbf{x})$ and introduce the generalization,

$$L_{RE,\lambda}(\hat{f}) = L_{RE}(\hat{f}) + \lambda L_{RE}^{(-)}(\hat{f}), \tag{18}$$

where $L_{RE}^{(-)}(\hat{f})$ is defined by replacing f and $\hat{f}$ in Eq. (17) with $-f$ and $-\hat{f}$ respectively. The value of λ is a tuning parameter that can be set according to the skew of the dataset. Data considered in this work contains extremes that skew positive. We therefore set $\lambda = 0.1$, so that the loss function focuses on predicting positive outliers.

2.5. Performance measures for regression with extreme events

Goodness of fit in the context of regression for extreme events is non-trivial to define. In Section 2.1 we outlined three criteria for a "good" predictor. That is, we seek models that accurately predict extreme events (in the sense that they may be used as a classifier), quantify extreme events, and yield accurate densities in the tails of p_y. In this section we present metrics for quantifying each of the three criteria and discuss potential shortcomings of our approach towards error analysis.

2.5.1. Accurate quantification of tails of $p_y(y)$

Finally, we seek models such that the push-forward density under the learned model is similar to the true density p_y. We are particularly interested in loss functions yielding densities that match the tails of the true density. Following [8,9] use the difference between logarithms of the two density functions over the intersection of their support. This is normalized by the size of

the intersection to penalize distributions that only intersect on a small domain. The metric is given by;

$$\mathbb{D}(p_y, \hat{p}_y) = \frac{1}{|\Omega(p_y, \hat{p}_y)|^2} \int_{\Omega(p_y,\hat{p}_y)} \left|\log(p_y(y)) - \log(\hat{p}_y(y))\right| \, dy, \tag{19}$$

where

$$\Omega(p_y, \hat{p}_y) \approx supp(p_y) \cap supp(\hat{p}_y). \tag{20}$$

and $\hat{p}_y$ is the density under the learned model. Since p_y and $\hat{p}_y$ are approximated from data, their support is unknown and behavior in low density regions extremely challenging to quantify. Unfortunately, $\mathbb{D}$ depends strongly on both of these quantities. We therefore approximate the support of each distribution as the interval covering the observed range of values. This is an underestimate of the true width, but allows for a consistent method of computing $\mathbb{D}$.

2.5.2. Accurate quantification of extreme events

Accurate quantification is indicated by models achieving low error on predictions from $\mathbf{x} \in X$ such that $f(\mathbf{x})$ is rare. We quantify this with the expected mean square error over the set of inputs corresponding to rare events. For example, consider the mean square error restricted to the set of events with $p_y(y) < \epsilon$;

$$MSE_\epsilon = \mathbb{E}_{\mathbf{x}}\left[e(\mathbf{x})^2 \middle| p_y(f(\mathbf{x})) \leq \epsilon \right], \tag{21}$$

for values of $\epsilon > 0$. Models that accurately quantify rare events should have low MSE_ϵ for $\epsilon \ll 1$. A more informative metric would condition on $p_y(f(\mathbf{x})) = \epsilon$, but computation of such a quantity requires a parametric model or smoother. We therefore do not include it in this work.

2.5.3. Accurate prediction of extreme events

Models trained for regression may be used along with some threshold value to function as classifiers. For a variety of extreme event rates, we track two metrics of classifier accuracy. Following [12], we use the extreme event rate dependent area under the precision recall curve given by,

$$\alpha(\omega; y, \hat{y}) = \int_{\mathbb{R}} s(1_{y\geq a}, 1_{\hat{y}\geq b}) \left| \frac{\partial}{\partial b} r(1_{y\geq a}, 1_{\hat{y}\geq b}) \right| db \tag{22}$$

where s and r are the precision and recall and $a = F_y^{-1}(1-\omega)$ where F_y^{-1} is the quantile function for y. This quantity has in fact been used to train models with lower dimensional parameter spaces. However, the use of distinct thresholds for y and $\hat{y}$ may be undesirable. Note for example that for any strictly increasing function g we have $\alpha(\omega; y, \hat{y}) = \alpha(\omega; y, g(\hat{y}))$. We therefore also consider a metric that uses the same threshold for both y and $\hat{y}$. The extreme event rate dependent F_1 score is given by,

$$F_1(\omega; y, \hat{y}) = F_1(1_{y\geq a}, 1_{\hat{y}\geq a}) \quad \text{where} \quad a = F_y^{-1}(1-\omega) \tag{23}$$

where the F_1-score is given by the harmonic mean of precision and recall.

In many practical settings, the consequences of false positive or negative predictions will be disproportionate. In these cases, the metrics given by Eq. (22) and (23) will not reflect the utility of the learned model and more setting specific metrics should be considered.

3. Results

In this section we present the results of using each of the loss functions discussed in Section 2 to two challenging supervised learning problems resulting from fluid dynamic systems with extreme events; the Kolmogorov flow at Reynolds number 40, and the flow around a square cylinder at Reynolds number 5000. Numerical simulations of the incompressible Navier–Stokes equations in each case were performed using the spectral element method implemented in Nek5000 [39,40], an open source Fortran code for incompressible fluids. Details on numerical simulation and data preparation are given in Appendix A. In both cases, target data y is centered and normalized to have zero mean and unit variance.

In each case we use LSTM networks implemented in tensorflow [41] and trained using Adam [42] with early stopping to avoid over-fitting. Inputs to the neural networks consist of short time series of Fourier modes or surface pressure up to some time t from which we seek to forecast energy dissipation or drag at a future time. Networks used for each example consist of a sequence of dense layers acting on each snapshot of the input time series independently, followed by an LSTM layer and another sequence of dense layers acting on the final LSTM output. Datasets are split into training (50%), validation (10%) and testing (40%). Each partition if formed from a contiguous set of samples so that phenomena observed in each are distinct. Further details on training, as well as details on network structure, layer sizes, and dimension are given in Appendix B. We report the metrics outlined in Section 2.5 evaluated on the portion of the data reserved for testing.

For each result we present data from twenty randomly initialized and trained networks. Plots show mean value of across all trials as a solid line and shade region between 10th and 90th percentiles, thus excluding the two highest and two lowest values.

3.1. Kolmogorov flow

We first consider two dimensional Kolmogorov flow at $Re = 40$. Dynamics follow the incompressible Navier–Stokes equations with a sinusoidal forcing term and periodic boundaries. Specifically,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} + \sin(k_f y)\mathbf{e}_1, \qquad \nabla\cdot\mathbf{u} = 0, \tag{24}$$

where $\mathbf{e}_1 = (1, 0)$, $k_f = 4$, and where we have boundary conditions $\mathbf{u}(2\pi, y, t) = \mathbf{u}(0, y, t)$ and $\mathbf{u}(x, 2\pi, t) = \mathbf{u}(x, 0, t)$. We are interested in this flow due to the behavior exhibited by the energy dissipation rate, given by a re-scaling of the enstrophy,

$$D(t) = \frac{\nu}{|\Omega|} \int_\Omega |\nabla\mathbf{u}(\mathbf{z}, t)|^2 \, d\mathbf{z}, \tag{25}$$

where $\nu = Re^{-1}$ is the viscosity and $\Omega = [0, 2\pi)^2$ is the computational domain. Solutions to Eq. (24) with the prescribed boundary conditions are known to exhibit large bursts in energy input and dissipation rate resulting from intermittent alignment of the velocity field with the external forcing [43]. A single snapshot of the velocity field in the x-direction, as well as the time series for the energy dissipation rate and its density are shown in Fig. 2.

The intermittent bursts apparent in Fig. 2 have been the subject of several previous works seeking to predict their occurrence in advance [12,29,43]. In particular, Farazmand and Sapsis [10] showed that the intermittent behavior could be explained through triadic interactions between Fourier modes. Here we consider the same problem, but seek to predict bursts in the energy dissipation using neural networks. The target (y) quantity is the energy dissipation rate $D(t)$, normalized to have zero mean and unit variance. We use the time varying magnitudes of the three Fourier modes identified in [10] as inputs; $\mathbf{x}(t) = (a_{0,k_f}(t), a_{1,0}(t), a_{1,k_f}(t))$. In order to ensure accurate statistical

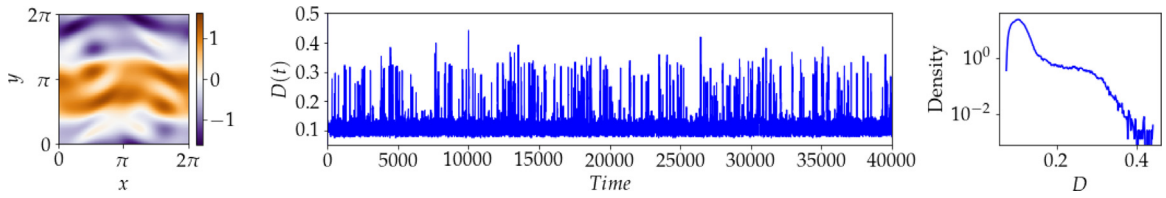

Fig. 2. Kolmogorov flow. Left: snapshot of x-velocity. Center: Time series for energy dissipation. Right: Empirical density function of energy dissipation for $t \in [0, 40,000]$.

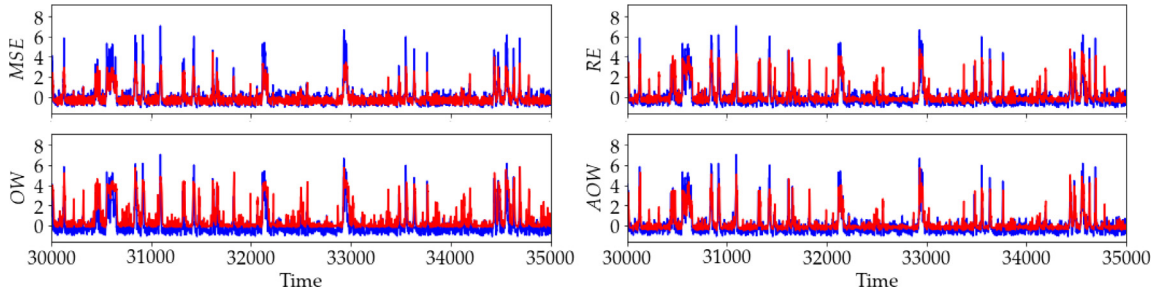

Fig. 3. Mean predictions for subset of test data for the Kolmogorov dataset at lead time $\tau = 6$. Blue time series is true $D(t)$ and red is mean prediction across 20 trained neural networks.

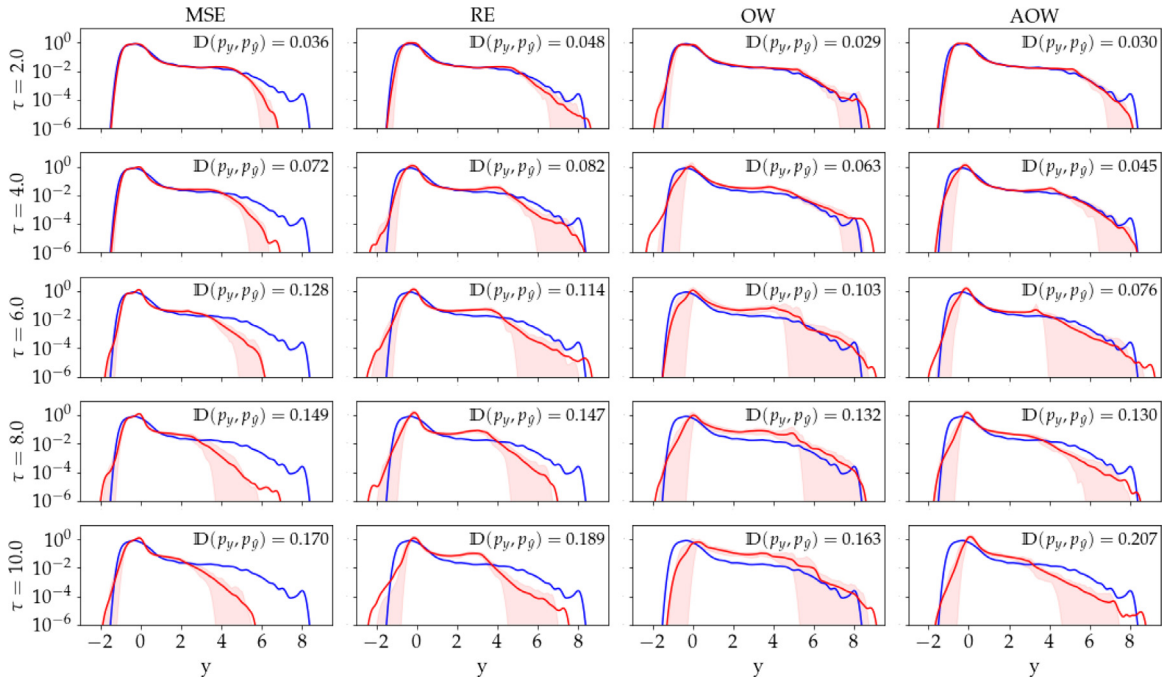

Fig. 4. Density for the Kolmogorov flow energy dissipation using true data (blue) and 20 realizations of neural networks trained with each loss function (red). Reported $\mathbb{D}$ is mean of Eq. (19).

descriptions of errors, a simulation of the Kolmogorov flow run for $4 \cdot 10^4$ time units.

We train neural network predictors of $D(t+\tau)$ from $\{\mathbf{x}(s) : s \leq t\}$ for several lead times τ. LSTM networks [31] are used to allow for historical data to assist in prediction. Further details on the neural network structure and training procedure may be found in Appendix B. For each network configuration and lead time, we train twenty networks from randomly initialized weights. A subset of the test data results is shown for a lead time of $\tau = 6$ in Fig. 3. The true value of $D(t)$ is shown in blue and the mean prediction across twenty networks for each loss function is shown in red. A cursory inspection reveals that L_{MSE} appears to underestimate the large fluctuations, though it does appear to predict their locations. Each of the three other loss functions appear to quantify the peaks more accurately, with L_{OW} having the most pronounced false positives.

Estimates of the densities p_y using test set values of $y_i = f(\mathbf{x}_i)$ and predictions $\hat{y}_i = \hat{f}(\mathbf{x}_i)$ with $\hat{f}$ trained using each of the loss functions considered in this work are shown in Fig. 4. The normalized difference between logs metric $\mathbb{D}$ is also shown alongside each loss function and lead time. At each lead time, the induced density using models trained with L_{MSE} has the highest fidelity in the core of the pdf, but these distributions dramatically underestimate the density in the high dissipation tail region. At a low lead time of $\tau = 2$, densities from models trained using L_{RE}, L_{OW}, and L_{AOW} perform approximately equally. For higher τ, L_{RE}

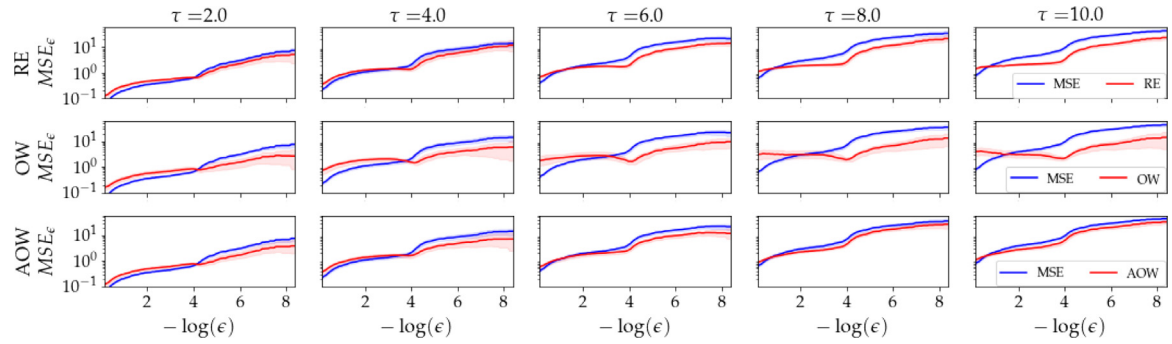

Fig. 5. Comparison of MSE_ϵ for models trained using L_{MSE} (blue) and extreme event specific loss functions (red) for the Kolmogorov dataset. Note that high $-\log(\epsilon)$ corresponds to rare events.

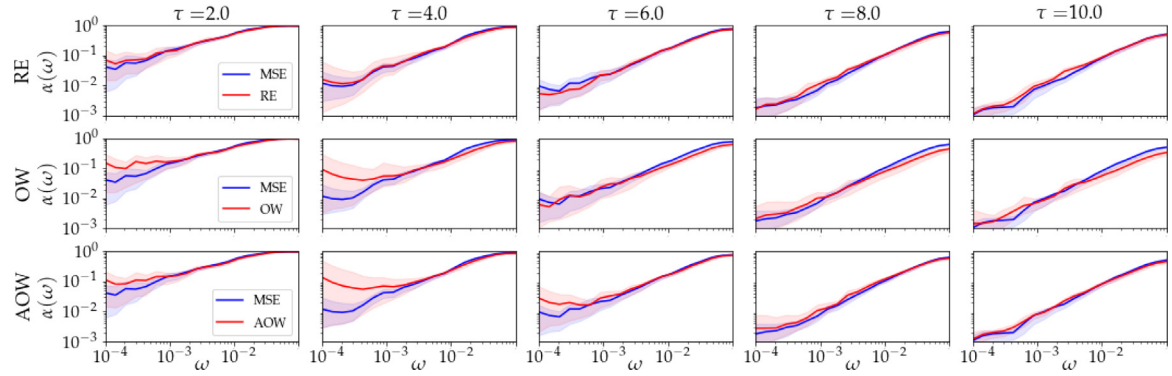

Fig. 6. Comparison of $\alpha(\omega)$ for models trained using L_{MSE} (blue) and extreme event specific loss functions (red) for the Kolmogorov dataset.

and L_{AOW} outperform L_{OW}, with the latter tending to overestimate the density positive outliers. Excepting $\tau = 10.0$, the minimal value of Eq. (19) is obtained by L_{AOW}. At $\tau = 10$ L_{MSE} achieves lower $\mathbb{D}\left(p_y, p_{\hat{y}}\right)$ than both L_{RE} and L_{AOW}. However, the estimated density function reveals significant underestimation of tail events. The low value of $\mathbb{D}\left(p_y, p_{\hat{y}}\right)$ is therefore explained by the restriction of the integral in Eq. (19) to the intersection of the supports of p_y and $p_{\hat{y}}$. At $\tau = 10$ L_{RE} and L_{AOW} both have significantly higher error in the tails of p_y than at $\tau = 8$ and thus higher $\mathbb{D}\left(p_y, p_{\hat{y}}\right)$.

The rare event error given by Eq. (21) is shown in Fig. 5. Each subplot includes error statistics for twenty networks trained with L_{MSE} in blue and networks trained using each of the specialized loss functions in red. In all cases, L_{MSE} has lower error when ϵ is small, which is unsurprising since this is the objective L_{MSE} minimizes. The difference is more pronounced when compared to L_{OW} or when τ is small. For rare events, each of the specialized loss functions has lower error.

The performance of the trained networks as classifiers for rare events is shown in Figs. 6 and 7. Fig. 6 shows that area under the precision recall curve as a function of extreme event rate. At low lead times and extreme event rates, L_{OW} and L_{AOW} show some improvement over MSE, but all methods are approximately equivalent in other cases. Note however, that $\alpha(\omega)$ allows for alternative thresholds to be applied to y and $\hat{y}$. Fig. 7 shows very clearly that across all cases the extreme event specific loss functions yield models that are more accurately able to distinguish extreme events when the same threshold is applied.

3.2. Flow around a square cylinder

Our second example considers the flow around a square cylinder at Reynolds number $Re = 5000$. The square has unit side-length and is positioned in a stream having unit inlet velocity and viscosity $\nu = 2 \cdot 10^{-4}$. The flow is characterized by extremely chaotic vortex shedding in the wake of the cylinder, shown in the left panel of Fig. 8 and large deviations in the forcing applied to the cylinder by the fluid. The drag coefficient is a non-dimensional quantity given by,

$$C_d(t) = \frac{2}{\rho u_\infty^2 C} \oint_{\partial S} (\boldsymbol{\tau}(t) - p(t)\mathbf{n}) \cdot \mathbf{e}_x \, ds \tag{26}$$

where $\boldsymbol{\tau}$, p, and $\mathbf{n}$ are the skin shear stress, pressure, and normal vector, and ∂S is the boundary of the square cylinder. ρ, u_∞, and C all have numerical value of 1 and are included only for their dimension. The time series of $C_d(t)$ for a simulation of length $T = 2 \cdot 10^4$ and empirical density function are shown in the center and right panels of Fig. 8.

We consider the problem of predicting $C_d(t + \tau)$, centered and normalized to unit variance, from 40 evenly spaced measurements of pressure on the surface of the cylinder, $P(t) \in \mathbb{R}^{40}$. The contributions of $\boldsymbol{\tau}$ and $p(t)$ to C_d are called the skin friction drag and pressure drag, respectively. In this example, pressure drag is several orders of magnitude larger than skin friction drag. Therefore, assuming a sufficiently dense distribution of pressure measurements prediction of C_d at zero lead time is proportional to the difference of average pressure on the front and back of the square. However, for $\tau > 0$ the problem rapidly becomes challenging.

We test of the loss functions proposed in Section 2 using an LSTM network described in further detail in Appendix B. Time series of the true drag coefficient as well as the mean prediction from twenty neural networks trained using each of the loss functions are shown in Fig. 9 for a lead time of $\tau = 1$. It is immediately clear that predictions made by networks trained with different loss functions exhibit substantially different behavior. Consistent with the Kolmogorov flow data, L_{MSE} substantially underestimates fluctuations, while L_{OW} has many false positives.

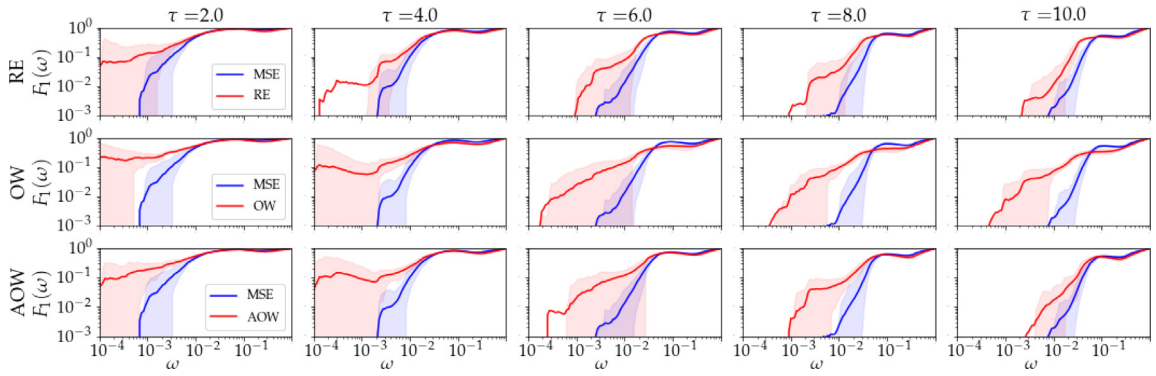

Fig. 7. Comparison of $F_1(\omega)$ for models trained using L_{MSE} (blue) and extreme event specific loss functions (red) for the Kolmogorov dataset. Lines indicate mean and shaded region is bounded by 10th and 90th percentiles.

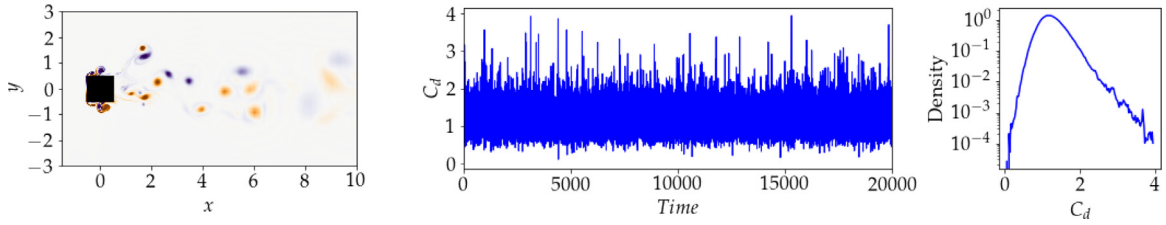

Fig. 8. Flow around a square cylinder at $Re = 5000$. Left: Snapshot of vorticity close to cylinder. Center: Time series for drag coefficient. Right: Empirical density function of drag coefficient.

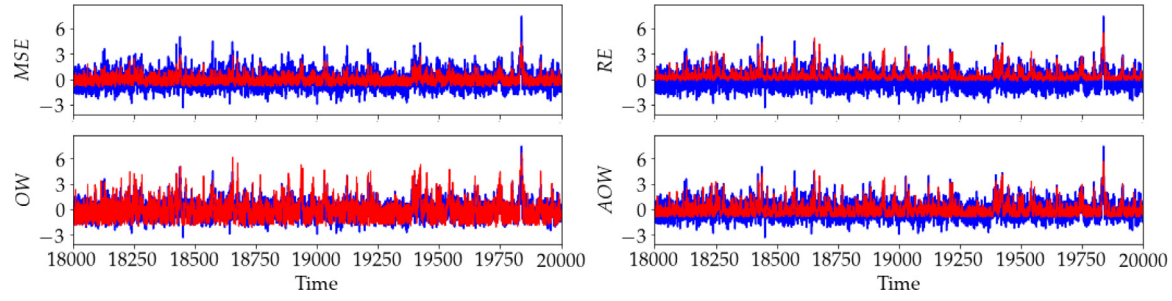

Fig. 9. True centered and normalized drag coefficient (blue) and mean time series predictions (red) for test set data from each neural network for the drag on a square cylinder. Results are for a lead time of $\tau = 1$.

Estimates of the density of y and of $\hat{y}$ formed from test set data for each loss function and several lead times are shown in Fig. 10. As expected, networks trained with L_{MSE} underestimate the likelihood of events in the tails of the distribution of y while L_{OW} overestimates these same events. L_{RE} and L_{AOW} generally both capture the correct tail behavior for extreme events skewing positive, with L_{AOW} achieving lower $\mathbb{D}$. We note that while L_{RE} performs poorly for quantifying p_y when $y < 0$, this may be a consequence of our choice of parameter λ, which weighed positive values ten times as much as negative. In each case, L_{RE} and L_{AOW} notable outperform L_{OW} which in turn outperforms MSE with regards to mean $\mathbb{D}$. Plots of MSE_ϵ are shown in Fig. 11. As expected loss functions tailored to extreme events are better able to quantify events occurring with a low probability (high $-\log(\epsilon)$), while underperforming L_{MSE} for more frequent events. This is consistent with expectations and supports the use of specialized loss functions for quantifying rare events.

Classification metrics for the square data are shown in Figs. 12 and 13. Fig. 12 provides evidence that challenges the utility of the extreme event specific loss functions. At a lead time of $\tau = 1$, L_{RE} slightly outperforms L_{MSE} in the low extreme event rate region. However, L_{MSE} has the highest $\alpha(\omega)$ in all other cases. This suggests that while networks trained L_{MSE} do a poor job at quantifying extreme events for this example, they are in fact able to do a better job than others at separating extreme from quiescent events given an appropriate pair of thresholds. However, this is not true when the same threshold is applied to each of y and $\hat{y}$. Fig. 13 shows that each of the extreme event specific loss functions outperforms L_{MSE} when the same threshold is used.

4. Discussion

In this work we have developed and evaluated several candidate loss functions for use in regression problems seeking to accurately quantify and predict extreme events. We have taken care to describe the motivation for each of the loss functions as an approximation of a continuous functional, allowing for greater intuition and providing a framework upon which further improvements may be made. The test cases used to evaluate the loss functions include the Kolmogorov flow, which has been studied extensively as a canonical example of a turbulent fluid flow exhibiting extreme events and the flow around a square cylinder at $Re = 5000$. This latter example has, to the best of our knowledge, not been studied in the context of extreme events. In

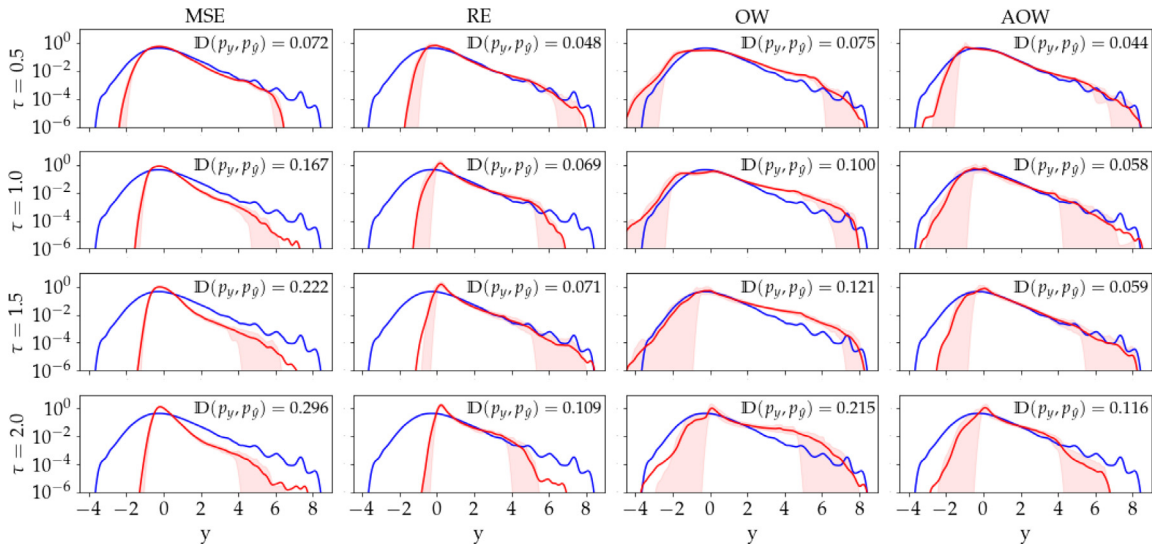

Fig. 10. Density functions for the drag on a square cylinder using true data (blue) and 20 realizations of neural networks trained with each loss function (red).

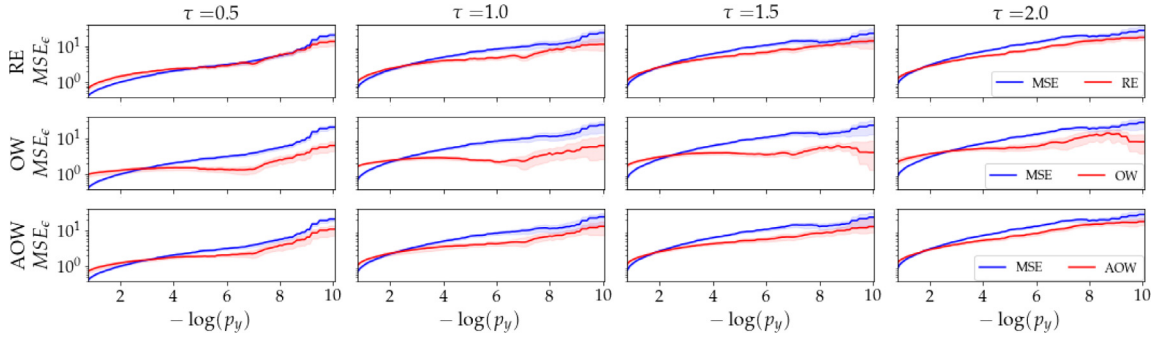

Fig. 11. Comparison of MSE_ϵ for models trained using L_{MSE} (blue) and extreme event specific loss functions (red) for the square dataset. Note, high $-\log(\epsilon)$ corresponds to rare events.

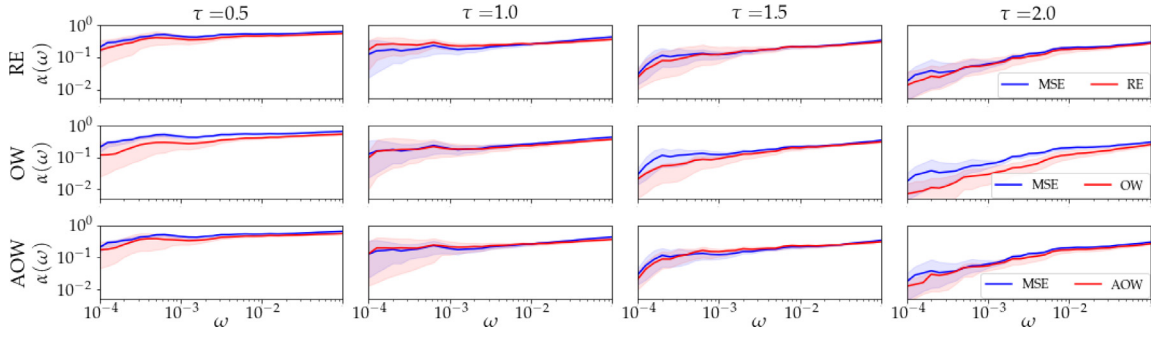

Fig. 12. Comparison of $\alpha(\omega)$ for models trained using L_{MSE} (blue) and extreme event specific loss functions (red) for the square dataset.

each case, the proposed loss functions yield models with significantly improved error in the tails of the output, more accurate estimates of the tail likelihoods, and improved classification when an equal threshold is applied to prediction and ground truth.

We have also provided evidence, however, that prediction and quantification of extreme events are distinct problems. Fig. 12 makes it clear that improved performance in extreme event quantification does not necessarily mean greater performance in classification, as measured by the area under the precision recall curve. However, this result requires distinct thresholds for what is considered an extreme event under f and $\hat{f}$. Thus, the area under the precision recall curve interprets $\hat{f}$ not as an interpolation of f but as a distinct metric used for classification.

The "best" loss function for a particular task will depend on the problem, dataset, and goals. Of those studied in this work, L_{MSE} and L_{RE} have the advantage of not requiring an estimate of p_y. The extent to which this proves troublesome will depend on the dimension of y, sample size, and on the distribution p_y. Tail events are most accurately quantified by L_{OW}, though at the cost of poor performance on events in the core of p_y. This is partially mitigated by L_{AOW}. Networks used in any application setting should be selected via cross-validation using a problem-appropriate metric.

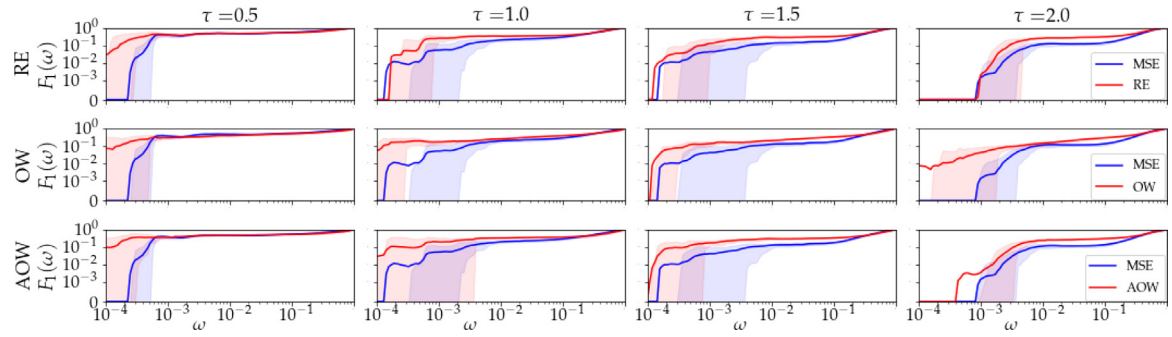

Fig. 13. Comparison of $F_1(\omega)$ for models trained using L_{MSE} (blue) and extreme event specific loss functions (red) for the square dataset.

An obvious challenge in the use of loss functions which include p_y is that the number of samples required for density approximation grows geometrically in the dimension of y. This requirement is particularly important in the context of quantifying probabilities for rare events such as those studied in this work, and may make extension of the proposed methods to higher dimensional problems challenging. We have focused on the case where y is scalar valued and thus avoided the issue. Problems in higher dimensions may require approximating density through some observable or via distance to neighbors along a low dimensional manifold via technique like diffusion maps [44], if such a manifold exists. The relative entropy loss does not require approximation of p_y, but does equate importance of a particular sample with its exponentiated magnitude. Thus, it may not be an ideal choice when the rare events of interest are not substantially different in magnitude from the core of the distribution. Further investigation of such problems would be an interesting research direction, but we consider it to be outside the scope of this work.

The proposed extreme event loss functions may be a poor choice for certain classes of prediction problems, even if the data has extreme events. Consider for example the task of learning a dynamic model for a system with extreme events using short time series or velocity data as in [13,45]. In this scenario, errors in the prediction phase are compounded and the higher error rate in the core of the distribution will render long term forecasts less accurate. This is in contrast to models trained with L_{MSE} where error accumulation will likely be more focused on rare events.

There exists a wide spectrum of functional forms used in machine learning that allows researchers to select or construct models they deem fit for a particular task. Less attention is paid to the choice of functional indicating the performance of those models, though some works have considered this question and proved its importance [13,14]. This may be due to the wide effectiveness of the often used mean square error, and its clear motivation as a maximum likelihood estimate given Gaussian error. However, as we have shown, performance on certain tasks is significantly improved with tailored loss functions. The present manuscript extends this important research direction.

CRediT authorship contribution statement

Samuel H. Rudy: Methodology, Software, Formal Analysis, Writing – original draft, Funding acquisition. **Themistoklis P. Sapsis:** Project administration, Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This material is based upon work supported by the National Science Foundation, United States of America (1902972), the Office of Navy Research, United States of America (N00014-21-1-2357), and the Army Research Office, United States of America (W911NF-20-1-0100) and (MURI W911NF-17-1-0306). Simulation of the flow around a square cylinder was run using the Extreme Science and Engineering Discovery Environment (XSEDE) [46] allocation TG-MTH210003. XSEDE is supported by National Science Foundation, United States of America under Grant No. ACI-1548562.

Appendix A. Details on datasets

Numerical simulations of the Kolmogorov flow and flow around a square cylinder were performed using Nek5000 [40], an open source Fortran based spectral element code for incompressible flow [39]. Time integration was carried out using a second order semi-implicit scheme described in [47]. Simulations were stabilized with a small degree of filtering as described in [48]. Mesh was generated using gmsh [49]. The Kolmogorov flow dataset uses 144 elements of order 7 and the flow around a square cylinder uses 1728 elements of order 7. The domain for the square cylinder flow extended from $x = -12$ to $x = 30$ and $|y| \leq 12$ with the square having sidelength 1 centered at the origin. Geometry and case files for rerunning simulations as well as files with numerical values of output data used in this work are available on GitHub at https://github.com/snagcliffs/EE_loss.

The mean and standard deviation of the energy dissipation for the Kolmogorov flow dataset used in this work are 0.116065 and 0.037559, which are respectively within 0.64% and 2.1% of those reported in [10]. The grid for the flow around a square cylinder was validated via comparison to a short simulation using a finer mesh. The mean drag coefficient on the interval $t \in [200, 2000]$ using 1728 spectral elements was found to be approximately 1.34% different from that using 4480 elements, which was deemed sufficiently resolved for the purpose of this work.

Appendix B. Neural networks details

Neural networks used in this manuscript were implemented in Python using the tensorflow library [41] as well as the Numba library [50]. Gaussian process models for p_y were initially trained using GPy [51], a Python library for Gaussian processes. Learned parameters were subsequently used to build non-trainable Gaussian process models in tensorflow, allowing for their use with neural network optimization tools. Source code for neural networks and scripts used to train examples used in this work are available on GitHub at https://github.com/snagcliffs/EE_loss.

Table 1
Parameters for neural networks.

Network	Pre-LSTM Dense	LSTM	Post-LSTM Dense	Input Size (dim.×time)	Input dt
Kolmogorov	4,8,16	32	16,8,4	3×20	1.0
Square Cyl.	4,8,16	16	16,8,4	40×5	0.1

Networks were built using a combination of dense layers and long-short-term-memory (LSTM) layers [31]. Details on the network structures and training are given in Table 1. Both networks were trained using a batch size of 1000 and used the swish activation function [52].

Neural networks are widely known to be prone to over fitting. To mitigate this, we used early stopping [53] and noise injection in the training data. Data was separated into disjoint temporally contiguous sets for training, validation, and testing using a (50/10/40%) split and training was stopped when validation set loss failed to yield a new minimum for 5 consecutive epochs. To avoid transient behavior, the initial 200 time units of each simulation were discarded. Noise injection has been shown to improve generalization performance [29,54], which may be due in part to its relation to Tikhonov regularization [55]. During the training procedure, we sampled random noise from a Gaussian distribution having standard deviation equal to 10% of the data and added it to the neural network input.

References

[1] David R. Easterling, J.L. Evans, P. Ya Groisman, Thomas R. Karl, Kenneth E. Kunkel, P. Ambenje, Observed variability and trends in extreme climate events: a brief review, Bull. Am. Meteorol. Soc. 81 (3) (2000) 417–426.
[2] P.K. Yeung, X.M. Zhai, Katepalli R. Sreenivasan, Extreme events in computational turbulence, Proc. Natl. Acad. Sci. 112 (41) (2015) 12633–12638.
[3] François M. Longin, The asymptotic distribution of extreme stock market returns, J. Bus. (1996) 383–408.
[4] Kristian Dysthe, Harald E. Krogstad, Peter Müller, Oceanic rogue waves, Annu. Rev. Fluid Mech. 40 (2008) 287–310.
[5] Themistoklis P. Sapsis, Statistics of extreme events in fluid flows and waves, Annu. Rev. Fluid Mech. 53 (2021) 85–111.
[6] Mohammad Farazmand, Themistoklis P. Sapsis, Extreme events: Mechanisms and prediction, Appl. Mech. Rev. 71 (5) (2019).
[7] Mustafa A. Mohamad, Themistoklis P. Sapsis, Sequential sampling strategy for extreme event statistics in nonlinear dynamical systems, Proc. Natl. Acad. Sci. 115 (44) (2018) 11138–11143.
[8] Themistoklis P. Sapsis, Output-weighted optimal sampling for Bayesian regression and rare event statistics using few samples, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 476 (2234) (2020) 20190834.
[9] Antoine Blanchard, Themistoklis Sapsis, Bayesian optimization with output-weighted optimal sampling, J. Comput. Phys. 425 (2021) 109901.
[10] Mohammad Farazmand, Themistoklis P. Sapsis, A variational approach to probing extreme events in turbulent dynamical systems, Sci. Adv. 3 (9) (2017) e1701533.
[11] Patrick J. Blonigan, Mohammad Farazmand, Themistoklis P. Sapsis, Are extreme dissipation events predictable in turbulent fluid flows? Phys. Rev. Fluids 4 (4) (2019) 044606.
[12] Stephen Guth, Themistoklis P. Sapsis, Machine learning predictors of extreme events occurring in complex dynamical systems, Entropy 21 (10) (2019) 925.
[13] N.A.K. Doan, W. Polifke, L. Magri, Short-and long-term predictions of chaotic flows and extreme events: a physics-constrained reservoir computing approach, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 477 (2253) (2021) 20210135.
[14] Di Qi, Andrew J. Majda, Using machine learning to predict extreme events in complex systems, Proc. Natl. Acad. Sci. 117 (1) (2020) 52–59.
[15] M.P. Brenner, J.D. Eldredge, J.B. Freund, Perspective on machine learning for advancing fluid mechanics, Phys. Rev. Fluids 4 (10) (2019) 100501.
[16] Steven L. Brunton, Bernd R. Noack, Petros Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2020) 477–508.
[17] Michele Milano, Petros Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. 182 (1) (2002) 1–26.
[18] Maziar Raissi, Paris Perdikaris, George 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 (2019) 686–707.
[19] Karthik Duraisamy, Gianluca Iaccarino, Heng Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. 51 (2019) 357–377.
[20] Abhinav Gupta, Pierre F.J. Lermusiaux, Neural closure models for dynamical systems, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 477 (2252) (2021) 20201004.
[21] Noah D. Brenowitz, Christopher S. Bretherton, Prognostic validation of a neural network unified physics parameterization, Geophys. Res. Lett. 45 (12) (2018) 6289–6298.
[22] Stephan Rasp, Michael S. Pritchard, Pierre Gentine, Deep learning to represent subgrid processes in climate models, Proc. Natl. Acad. Sci. 115 (39) (2018) 9684–9689.
[23] Christopher Irrgang, Niklas Boers, Maike Sonnewald, Elizabeth A Barnes, Christopher Kadow, Joanna Staneva, Jan Saynisch-Wagner, Towards neural earth system modelling by integrating artificial intelligence in earth system science, Nat. Mach. Intell. 3 (8) (2021) 667–674.
[24] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, George Em Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[25] Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, Karen Willcox, Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems, Physica D 406 (2020) 132401.
[26] Bethany Lusch, J. Nathan Kutz, Steven L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature Commun. 9 (1) (2018) 1–10.
[27] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, Steven L. Brunton, Data-driven discovery of coordinates and governing equations, Proc. Natl. Acad. Sci. 116 (45) (2019) 22445–22451.
[28] A.J. Majda, D.W. McLaughlin, EG1431687 Tabak, A one-dimensional model for dispersive wave turbulence, J. Nonlinear Sci. 7 (1) (1997) 9–44.
[29] Anna Asch, Ethan J. Brady, Hugo Gallardo, John Hood, Bryan Chu, Mohammad Farazmand, Model-assisted deep learning of rare extreme events from partial observations, Chaos 32 (4) (2022) 043112.
[30] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
[31] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[32] Joseph T. Chang, David Pollard, Conditioning as disintegration, Stat. Neerl. 51 (3) (1997) 287–317.
[33] Haibo He, Edwardo A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[34] Carl Edward Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[35] Andrew McHutchon, Differentiating gaussian processes, Cambridge (Ed.), 2013.
[36] Larry Wasserman, All of Nonparametric Statistics, Springer Science & Business Media, 2006.
[37] Bradley Efron, Nonparametric standard errors and confidence intervals, Canad. J. Statist. 9 (2) (1981) 139–158.
[38] David Siegmund, Importance sampling in the Monte Carlo study of sequential tests, Ann. Statist. (1976) 673–684.
[39] Anthony T. Patera, A spectral element method for fluid dynamics: laminar flow in a channel expansion, J. Comput. Phys. 54 (3) (1984) 468–488.
[40] James W. Lottes Paul F. Fischer, Stefan G. Kerkemeier, nek5000 Web page, 2008, http://nek5000.mcs.anl.gov.
[41] Martín Abadi, et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org.
[42] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[43] Mohammad Farazmand, An adjoint-based approach for finding invariant solutions of Navier–Stokes equations, J. Fluid Mech. 795 (2016) 278–312.
[44] Ronald R. Coifman, Stéphane Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. 21 (1) (2006) 5–30.
[45] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, Themistoklis Sapsis, Data-assisted reduced-order modeling of extreme events in complex dynamical systems, PLoS One 13 (5) (2018) e0197704.
[46] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, et al., XSEDE: accelerating scientific discovery, Comput. Sci. Eng. 16 (5) (2014) 62–74.
[47] P.F. Fischer, Implementation considerations for the OIFS/characteristics approach to convection problems, Argonne National Laboratory, 2003.
[48] Paul Fischer, Julia Mullen, Filter-based stabilization of spectral element methods, C. R. de L'académie Des Sciences-Series I-Mathematics 332 (3) (2001) 265–270.

[49] Christophe Geuzaine, Jean-François Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities, Internat. J. Numer. Methods Engrg. 79 (11) (2009) 1309–1331.
[50] Siu Kwan Lam, Antoine Pitrou, Stanley Seibert, Numba: A llvm-based python jit compiler, in: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, pp. 1–6.
[51] GPy, GPy: A Gaussian process framework in python, since 2012, http://github.com/SheffieldML/GPy.
[52] Prajit Ramachandran, Barret Zoph, Quoc V. Le, Searching for activation functions, 2017, arXiv preprint arXiv:1710.05941.
[53] Nelson Morgan, Hervé Bourlard, Generalization and parameter estimation in feedforward nets: Some experiments, Adv. Neural Inf. Process. Syst. 2 (1989) 630–637.
[54] Guozhong An, The effects of adding noise during backpropagation training on a generalization performance, Neural Comput. 8 (3) (1996) 643–674.
[55] Chris M. Bishop, Training with noise is equivalent to Tikhonov regularization, Neural Comput. 7 (1) (1995) 108–116.