Discovering and forecasting extreme events via active learning in neural operators

Ethan Pickering[1,*], George Em Karniadakis[2], and Themistoklis P. Sapsis[1,*]

[1]Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02138, USA

[2]Division of Applied Mathematics, Brown University, Providence, RI 02906, USA

April 7, 2022

Abstract

Extreme events in society and nature, such as pandemic spikes or rogue waves, can have catastrophic consequences. Characterizing extremes is difficult as they occur rarely, arise from seemingly benign conditions, and belong to complex and often unknown infinite-dimensional systems. Such challenges render attempts at characterizing them as moot. We address each of these difficulties by combining novel training schemes in Bayesian experimental design (BED) with an ensemble of deep neural operators (DNOs). This model-agnostic framework pairs a BED scheme that actively selects data for quantifying extreme events with an ensemble of DNOs that approximate infinite-dimensional nonlinear operators. We find that not only does this framework clearly beat Gaussian processes (GPs) but that 1) shallow ensembles of just *two* members perform *best*; 2) extremes are uncovered regardless of the state of initial data (i.e. with or without extremes); 3) our method eliminates "double-descent" phenomena; 4) the use of batches of suboptimal acquisition points compared to step-by-step global optima does not hinder BED performance; and 5) Monte Carlo acquisition outperforms standard minimizers in high-dimensions. Together these conclusions form the foundation of an AI-assisted experimental infrastructure that can efficiently infer and pinpoint critical situations across many domains, from physical to societal systems.

1 Introduction

The grand challenge of predicting disasters remains an extremely difficult and unsolved problem [1]. Disasters, such as pandemic spikes, wildfires or rogue waves[1], are uniquely challenging to quantify. This is because they are both *rare* and arise from an *infinite* set of physical conditions [4]. The proposition of predicting extremes is analogous to *finding a catastrophic needle in an infinite-dimensional haystack*. This calls for methods that can both *discover* extreme events and encode *physical* phenomena into their modeling strategy. We present a Bayesian-inspired experimental design (BED) approach, described in detail in figure 5, that addresses both challenges by combining a probabilistic "discovery" algorithm [5] with a deep neural operator designed to approximate physical systems [6].

Discovery of extremes is often simplified by distilling complex systems to their governing input variables and relevant output variables. Within this interpretation, quantification of extremes has historically taken the form of importance sampling, which uses a biasing distribution to identify regions of the input space that exhibit extreme values [7, 8]. Unfortunately, these techniques often require additional and challenging considerations for accurate results [9, 10, 11] and are static, lacking an ability to adjust to new information gained through experiments. BED provides a dynamic approach that learns from acquired data before

*Co-corresponding authors: pickering@mit.edu, sapsis@mit.edu

[1]Rogue waves are rare, giant waves that pose a danger to ships and offshore structures. The largest wave on record was 25.6 meters and hit the Draupner oil platform in the North Sea on January 1, 1995 [2]. The "most extreme" wave, three times the size of surrounding waves, was 19.5 meters and observed off the coast of Ucluelet, British Columbia, on November 17, 2020 [3]

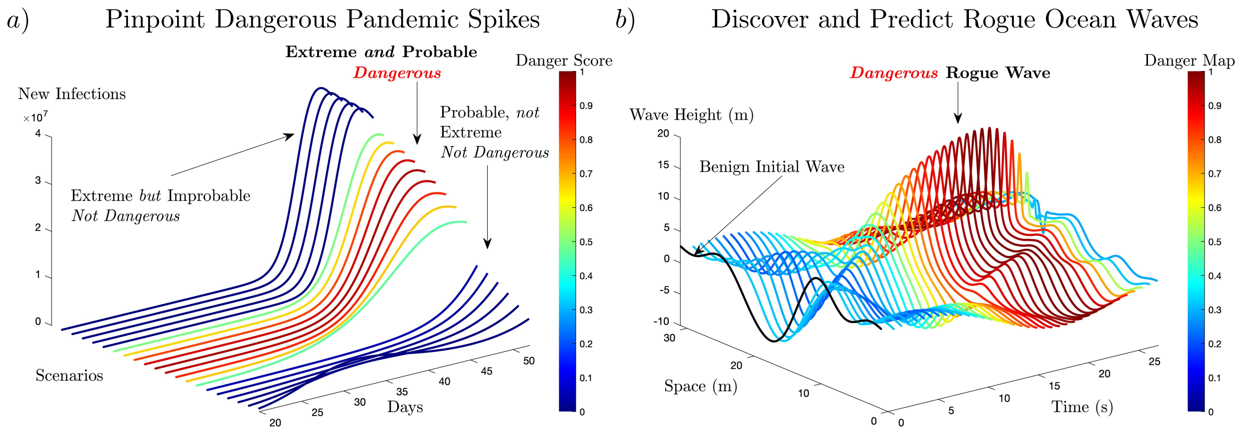

Figure 1: **Inference of diverse extreme phenomena, from pandemic spikes to rogue ocean waves.** Our framework pinpoints the most dangerous (i.e., probable and extreme) pandemic scenarios (left) and discovers rogue waves (right). The most dangerous pandemic scenarios are pinpointed by inferring the number of new infections in time for a plurality of infection rate hypotheses. See figure 2 for more details. Rogue waves are discovered and quantified for future prediction by uncovering the probable wave conditions that non-linearly interact in time to emit rogue waves over three times their original size. We show one example of this phenomena here and refer to figure 4 for more details on discovering these waves.

selecting new and intriguing input-output data. However, BED has not been a popular choice in rare-event characterization due to a lack of 1) surrogate models that accurately generalize infinite-dimensional systems, and 2) appropriately defined acquisition functions for selecting extreme data.

Deep neural operators (DNO), such as DeepONet [6], are built specifically for handling infinite-dimensional systems and provide the ideal surrogate model for characterizing extremes. Unlike other ML approaches, such as Gaussian Process Regression (GP), which map parameterizations of physical phenomenon, DNOs directly map *physical*, infinite-dimensional functions to *physical*, infinite-dimensional functions. This leads to drastic improvements in generalization to unseen data in high dimensions. Additionally, the neural network backbone of DNOs mean they are intrinsically amenable to big data, unlike GPs which scale as the third power of data size [12, 13]. However, DNOs utility for Bayesian experimental design is an open question as DNOs do not explicitly provide a measure of uncertainty. We propose and show the efficacy of using an ensemble of DNOs for uncertainty quantification and BED. Although much of the literature is skeptical of the generality of ensembles to provide uncertainty estimates, recent viewpoints, [14] and notably [15], have argued that DNN ensembles provide a very good approximation of the posterior. Our results support this perspective.

Appropriately defined acquisition functions for uncovering extreme behavior are just as critical as the chosen surrogate model. Recently, [16], in concert with several other works [17], [18], and [5], introduced a class of probabilistic acquisition functions specifically designed for discovering extreme events. By combining statistics of the input space along with statistics deduced from the surrogate model, the method can account for both the importance of the output relative to the input. This approach significantly reduces the number of input samples required to characterize extreme phenomena.

Equipped with these acquisition functions and an ensemble of DNOs, we will search high-dimensional systems to discover extreme rogue waves and pinpoint dangerous pandemic scenarios, the two classes of representative problems we consider herein.

2 Results

Figure 1 presents the key implications of our results, diagnosing the most dangerous future pandemic scenarios (left) and discovering unassuming waves that lead to dangerous rogue waves (right). In each case, different

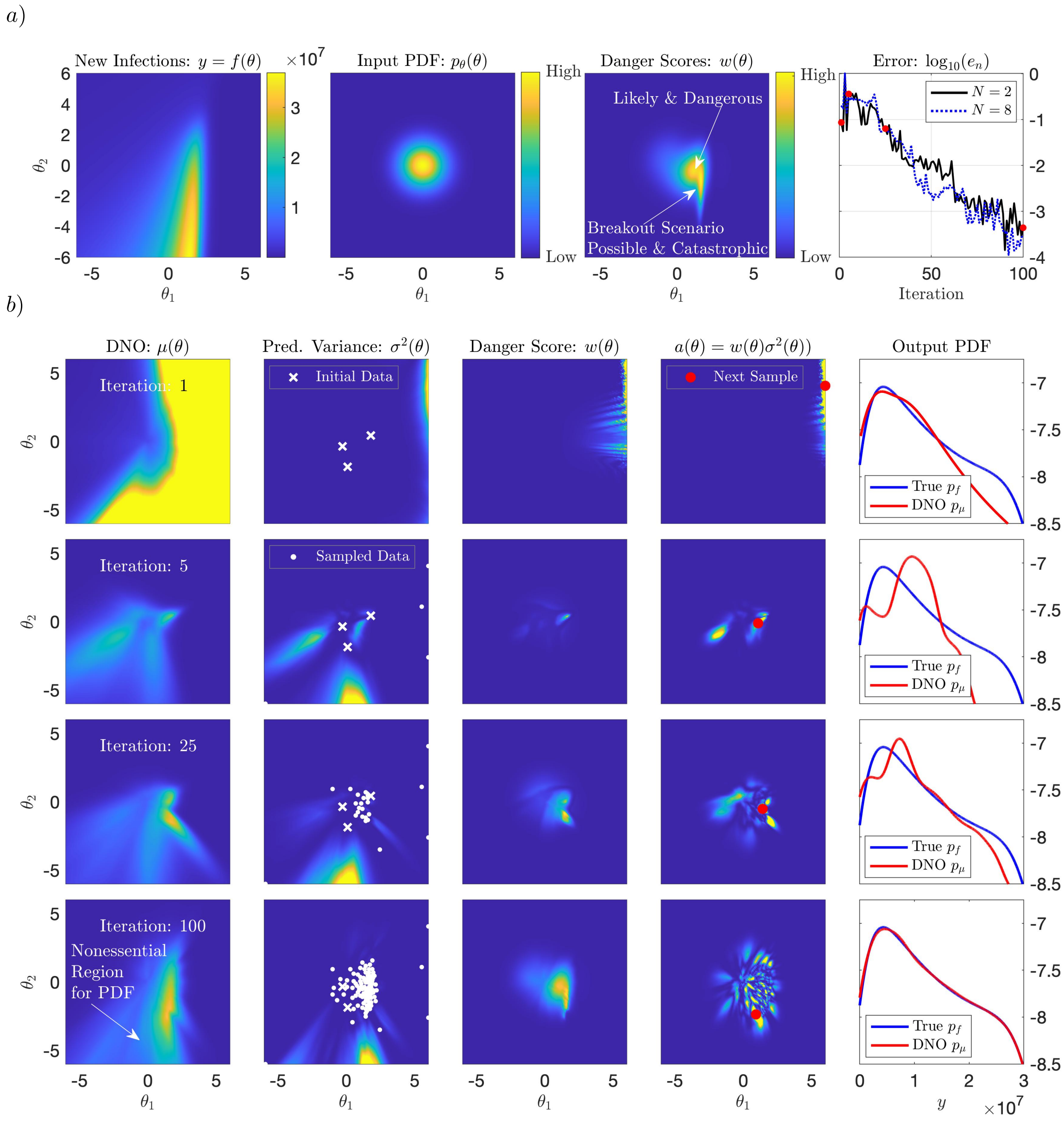

Figure 2: **Nailing the Tail: Accurate PDF and danger map convergence in 100 samples.** *a*) From left to right, the full deterministic response of new infections, $f(\boldsymbol{\theta})$, with respect to the two random parameters, θ_1 and θ_2, the probability distribution of the random parameters θ_1, θ_2, the underlying danger scores, $w(\boldsymbol{\theta})$, and the $\log_{10}$ of the log-pdf error for the experiment performed in *b*) using $N = 2$. The red circles indicate the iterations shown in *b*) as well as additional results for a case with $N = 8$ ensembles. *b*) One experiment of the 2D stochastic SIR model using three initial points and iterated 100 times. The rows give the iteration number, 1, 5, 25, and 100 from top to bottom, respectively. The columns, from left to right, are the DNO approximation of the objective function, $\mu(\boldsymbol{\theta})$, given the training points (initial + samples), the points sampled (where initial points are white X and sampled points are white circles) in the 2D parameter space and the predictive variance $\sigma^2(\boldsymbol{\theta})$, the calculated danger scores $w(\boldsymbol{\theta})$, the acquisition values $a(\boldsymbol{\theta})$ with the next acquisition point denoted by a red circle, and the DNO approximated and true output PDFs. Animation links presenting all 100 iterations for $N = 2$ (here) and $N = 8$ (here).

scenarios are tied to a "danger score" or *likelihood ratio*, as proposed by [5],

$$w(\boldsymbol{\theta}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\theta})}{p_\mu(\mu(\boldsymbol{\theta}))}, \tag{1}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ is the probability density function (PDF) of the random variable input, $\boldsymbol{\theta}$, and $p_\mu(\mu(\boldsymbol{\theta}))$ is the posterior PDF of the mean, $\mu(\boldsymbol{\theta})$, conditioned on the input,

$$p_\mu(\mu(\boldsymbol{\theta})) = \int_{\boldsymbol{\theta}} p_\mu(\mu|\boldsymbol{\theta}) p_{\boldsymbol{\theta}}(\boldsymbol{\theta})|\mathrm{d}\boldsymbol{\theta}. \tag{2}$$

The likelihood ratio appropriately balances events that are likely and those that are extreme, hence it provides a danger score for any given event. As denoted for the pandemic model in figure 1, small danger scores are attributed to events that are either implausible or not extreme, while large danger scores relate to those that are both likely and extreme. However, any system's danger score requires knowledge of the posterior mean, $p_\mu(\mu(\boldsymbol{\theta}))$, which is generally unknown and must be learned. Our approach efficiently learns this underlying distribution through dynamic application of the danger score with Bayesian experimental design and deep neural operators.

2.1 Pinpointing Dangerous Pandemic Scenarios

Figure 2 *a*) and *b*) demonstrate how the proposed framework efficiently learns the underlying output distribution of infections for a stochastic pandemic model via dynamic and sequential application of danger scores. The pandemic model is a simple Susceptible, Infected, Recovered (SIR) model with a two-dimensional stochastic infection rate (see Appendix A). We assess success as the log-PDF error in figure 2 *a*) last column, between the true output distribution and the approximated distribution from the trained DNO, figure 2 *b*) last column. See Section 4.2.3 for details on computing the log-PDF error metric.

Despite an initialization of only three data points in the parameter space, our framework quickly identifies the key regions of dynamical relevance and accurately recovers the significant properties of the underlying system. Using an ensemble of just two DNOs (see Appendix D for DNO implementation and section 4.1.1 for our application of ensemble methods), the algorithm iteratively provides an estimation of the underlying map (for computing $p_\mu(\mu(\boldsymbol{\theta}))$), variance, $\sigma^2(\boldsymbol{\theta})$, and a danger score, $w(\boldsymbol{\theta})$. Together, the danger score and variance create the acquisition function, $a(\boldsymbol{\theta}) = w(\boldsymbol{\theta})\sigma^2(\boldsymbol{\theta})$, that identifies the point within the parameter space with the greatest potential for learning the true output PDF. With the addition of each point, all fields dynamically change and bring the true and approximated output PDFs within greater agreement. By iteration 100, the danger scores have converged and the approximated output PDF is in error of less than 10^{-3}. It is from this final danger score map that we derive the pandemic scenarios of figure 1 and note that this map includes both dangerous pandemic spike scenarios and a breakout scenario with catastrophic consequences. Additionally, figure 2 *a*) last column, shows that increasing the ensemble size to $N = 8$ provides little to no advantage.

The critical aspect of this approach is the algorithm's reduction of a large parameter space to local regions of danger. Only regions that provide significant contributions to the output PDF are considered. Iteration 100 in figure 2 b) underscores this behavior. The algorithm has accurately reconstructed the output PDF, yet it ignores a large region of high magnitude. This region, and the remaining unexplored regions, provides negligible information and is neglected. This property is crucial for all systems where resources for experiments or simulation are limited or costly and it permits a significant reduction in training/acquired data as system complexity and dimensionality increases.

2.2 Discovering and Predicting Rogue Waves

We now scale the method to higher dimensions, larger datasets, and on a significantly more complex and nonlinear system: a dissipative version of the nonlinear Schrödinger equation used to model rogue waves (Appendix B). Figure 3 a)-c) shows that the DNO-BED framework, using the likelihood-weighted uncertainty sampling (US-LW, $a(\boldsymbol{\theta}) = w(\boldsymbol{\theta})\sigma^2(\boldsymbol{\theta})$), is superior for approximating the output PDF of the danger map (see section 4.2.3 for NLS danger map definition) than using GPs (detailed in Appendix C) or other common BED

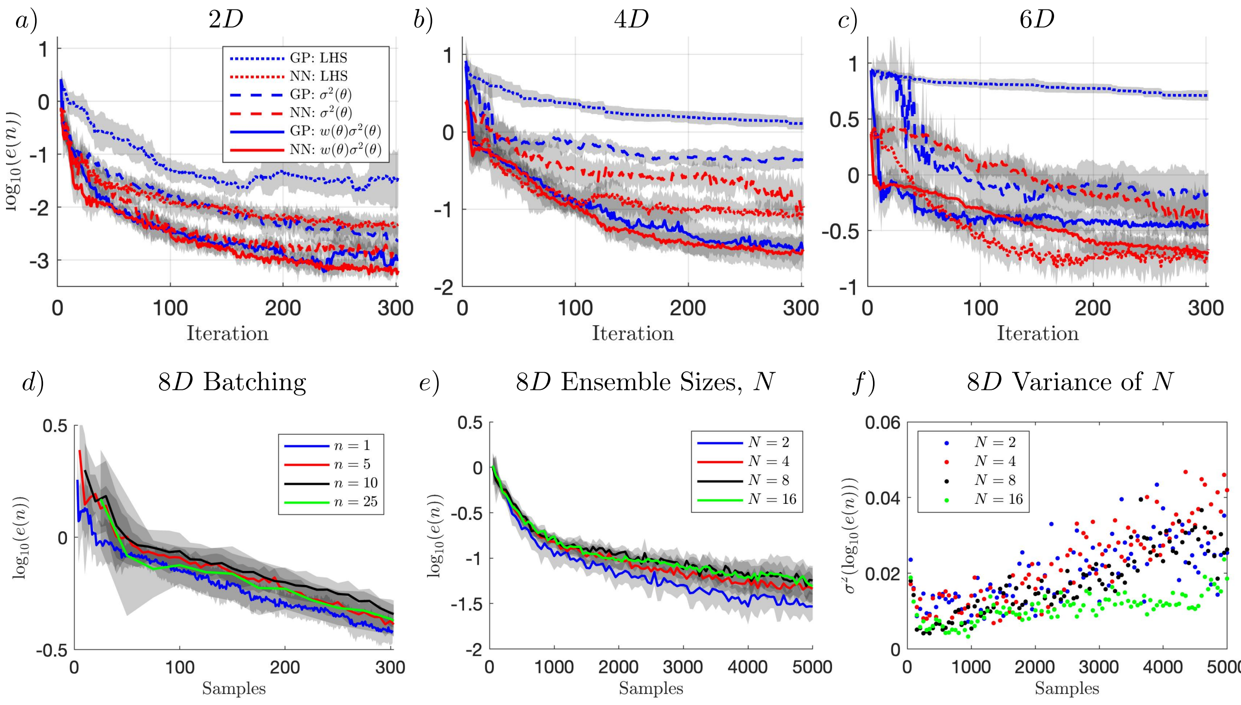

Figure 3: **Accelerated convergence with DNOs plus extreme acquisition functions regardless of dimensionality.** $a) - c)$ Median log-PDF of 10 independent and randomly initialized experiments, where shaded regions denote one standard deviation for each case (i.e. $\pm\sigma(\epsilon)$), considering three acquisition functions and GP and DNO surrogate models. $a) - c)$ are for $2D$, $4D$ and $6D$ initial conditions, while $d) - f)$ represent results from $8D$ initial conditions. **Parallel acquisition and shallow ensembles bring computational efficiency without performance loss.** $d)$ provides errors related to different batch sizes per iteration, $e)$ gives performance differences relative to the DNO ensemble size (100 iterations at batch size $n = 50$), and $f)$ presents the variance in log-PDF error of $e)$.

sampling strategies, such as uncertainty sampling (US, $a(\boldsymbol{\theta}) = \sigma^2(\boldsymbol{\theta})$) and latin hypercube sampling (LHS). This is highlighted as dimensionality (i.e., complexity, details on dimensionality provided in Appendix B) is increased from $2D$ to $6D$, where GPs begin to break down.

Figure 3 $d)$ and $e)$ provide two major implications for reducing the framework's computational cost in high dimensions: neither batching samples via parallel selection of multiple acquisition points, $d)$, nor use of shallow ensembles of only $N = 2$ members, $e)$, hinders performance. Computed at $8D$, figure 3 $d)$ shows that regardless of choosing 1,5,10, or even 25 points per iteration, over 300 samples, does not result in a loss of performance with our framework. Further, we show in Appendix F that the use of Monte Carlo methods are substantially more efficient for identifying optimal acquisition points than standard PYTHON optimizers. See Section 4.4 for details on batching implementation. These observations are a critical result for scaling our framework, as more complex systems inevitably will require more data.

Unexpectedly, figure 3 $e)$ shows that not only do *just two ensemble members, $N = 2$, perform well, they consistently outperform larger ensembles, $N > 2$.* This result appears to disagree with the hypothesis that a larger set of ensembles would provide uncertainty estimates with greater fidelity, leading to better performance of our sequential search methods. Clearly, this is not the case from our results, yet neither can it be that $N = 2$ ensembles provide a predictive variance of greater fidelity than $N = 16$. Figure 3 $f)$ permits both concepts to be true. It shows that the greater the ensemble size, the smaller the variance between the error trajectories of independent experiments. This observation agrees with the idea that larger ensembles lead to a higher-fidelity predictive variance, but that greater fidelity leads to *consistency* rather than *performance* for this sequential search technique. We believe using small N imposes a greedy search, in a similar fashion to Thompson sampling [19]. Regardless, the consistent observation that $N = 2$ is not only viable, but perhaps preferable, has massive implications for minimizing computational costs for ensemble approaches.

2.2.1 Rogue Wave Discovery in $20D$

Equipped with the computational advantages of $N = 2$ ensemble members and large batch sizes ($n_b = 50$), figure 4 $a) - b)$ shows that even at $20D$ our approach can recover the true output PDF. The other acquisition functions not only perform poorly, but perform worse as more data is acquired. This phenomenon is known as "double descent", and many researchers have observed this behavior throughout machine learning procedures, from classification to regression problems [20]. To our knowledge, this is the first observation of double descent arising from the tails of the log-PDF error. Double descent is a product of over-fitting. More data and greater complexity results in an over-parameterization of the provided data. Temporarily, this leads to inferior generalization until providing the surrogate model with sufficiently larger datasets.

The proposed acquisition function clearly avoids double descent. We believe this is because of the efficient selection of points that significantly contribute to the observed dynamics of the system. Unlike US-LW, the points chosen by LHS and US methods are not inherently important to recovering the true PDF and therefore induce misleading complexity to the underlying regression task. This observation further underscores the value of our acquisition function, as it systematically prevents over-fitting and unwarranted model complexity.

Figure 4 $a) - b)$ also show that regardless of the origin of initial conditions, containing extremes (chosen by LHS) or without extremes (from the prior, p_θ), the method achieves similar error metrics. Figure 4 $c)$ and $d)$ present the physical manifestations of the $20D$ initial conditions that include extremes and do not. Despite these clearly different sets of initial data, the algorithm uncovers dangerous benign conditions that lead to extremes.

3 Discussion

The discovery, characterization, and ultimate prediction of extreme events remains a grand challenge across application domains. Presented in figure 5, we propose an AI-assisted, model-agnostic infrastructure that paves the way for efficient and robust inference of these critically important events. By combining deep neural operators, Bayesian experimental design, and tailored acquisition functions, we create a framework that can accurately discover, characterize, and predict extreme events in high-dimensional nonlinear systems ranging from societal to physical. We demonstrate the efficacy of this framework here by efficiently discovering and predicting rogue waves, reducing the number of samples from 10^7 to 10^2, and accurately pinpointing the most dangerous pandemic spikes.

This approach is not only superior to other techniques, namely Gaussian Processes (GPs), but comes with several expected and unexpected, yet preferable, consequences. As expected, we show that the use of DNOs brings both greater generalization accuracy and a propensity for extremely large datasets. Unexpectedly, we find that shallow ensembles of only *two* ensemble members perform remarkably well and that the use of batches of optimal sample points acquired in parallel, compared to step-by-step, or serial, acquisition, does not hinder BED performance. We also find that Monte Carlo approaches for selecting these optimal points perform better than standard optimizers, which routinely fall victim to the highly non-convex nature of high-dimensional optimization. These results bring crucial real-world benefits to our approach. Batching permits parallel acquisition of data, while small ensembles keep computational costs tractable.

Our method also eliminates "double descent" phenomena and achieves accurate extreme event quantification *regardless* of the state of initial data. Both observations underscore the importance of sampling the *most informative data rather than more data.* Double descent is avoided as the acquisition function only selects points that significantly contribute to the observed dynamics of the system, while ignoring data that would bring added and unnecessary complexity. On the other hand, the method is robust to the initial data provided and quickly directs attention to the most informative samples whether or not extremes have been observed.

The framework also provides modularity for the chosen surrogate models and acquisition functions, so long as the parameter and function spaces are appropriately defined. Critically, we separate the two. Regression is performed in the functional space, while the search algorithm is performed in the parameter space via a forward DNO coupling (i.e., through a linear transfer function from $\boldsymbol{\theta} \rightarrow u(x)$). Any arbitrary neural architecture leveraging this distinction may be implemented, such as standard feed-forward NNs, Fourier

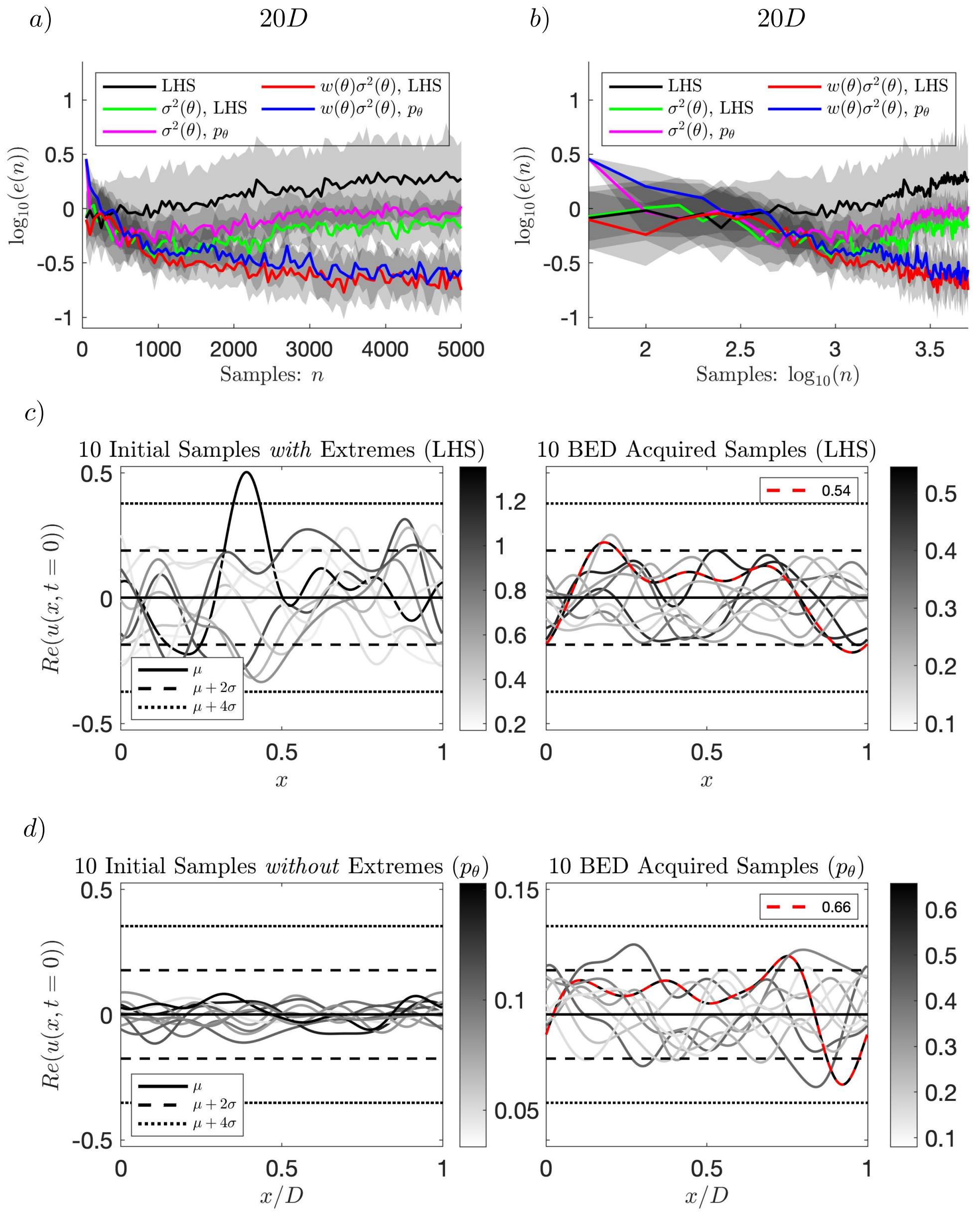

Figure 4: **Robustness to dimensionality and initial data, i.e., with or without extremes.** *a*) The log-PDF errors for three acquisition functions (LHS, US, and US-LW) for the $20D$ problem, with initial points sampled via LHS and the prior p_θ for US and US-LW. *b*) provides the same results in log-log form. *c*) gives the real value of 10 complex LHS initial input functions $u(x, t = 0)$, shaded with their corresponding output value $||Re(u(x, t = \tau))||_\infty$ (left) and 10 DNO+BED acquired functions, via US-LW at iteration 97 (right). *d*) gives 10 initial input functions sampled from the prior distribution p_θ (left) and the 10 acquired function at iteration 74 (right).

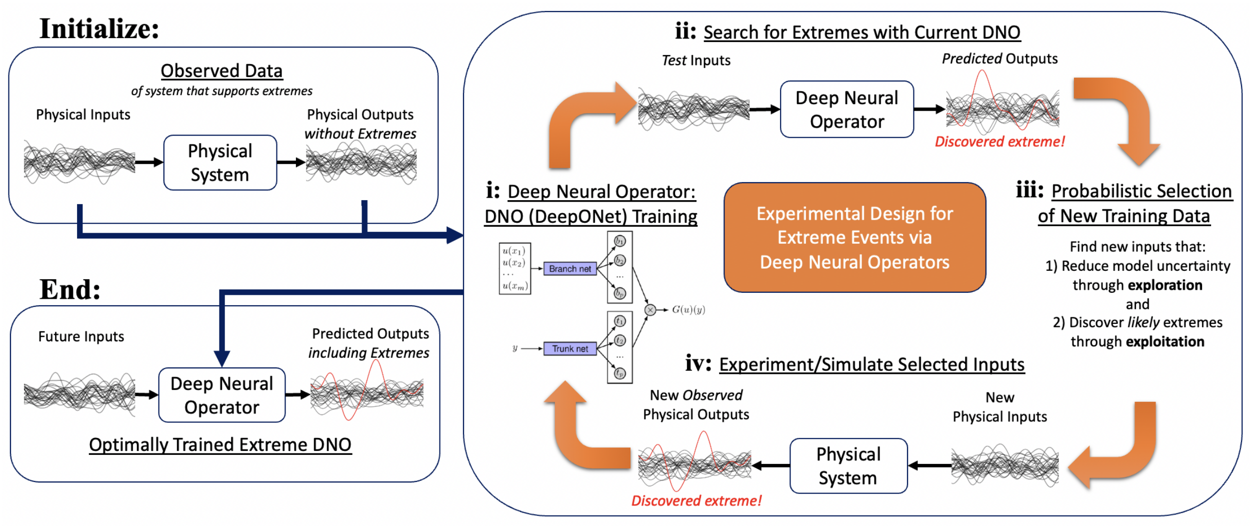
Initialize:
Observed Data
of system that supports extremes
Physical Inputs
Physical Outputs without Extremes
Physical System
End:
Future Inputs
Predicted Outputs including Extremes
Deep Neural Operator
Optimally Trained Extreme DNO
ii: Search for Extremes with Current DNO
Test Inputs
Predicted Outputs
Deep Neural Operator
Discovered extreme!
i: Deep Neural Operator: DNO (DeepONet) Training
Branch net
Trunk net
Experimental Design for Extreme Events via Deep Neural Operators
iii: Probabilistic Selection of New Training Data
Find new inputs that:
1) Reduce model uncertainty through exploration
and
2) Discover likely extremes through exploitation
iv: Experiment/Simulate Selected Inputs
New Observed Physical Outputs
New Physical Inputs
Physical System
Discovered extreme!

Figure 5: **Efficient and robust DNO+BED framework for discovering and quantifying extremes.** An overview of the proposed Bayesian experimental design framework with deep neural operators and novel acquisition functions for discovering extremes. *Initialize)* with a set of observed physical input-output pairs, retained in their functional form. *i)* Pass functions to an ensemble of DNOs to learn sparse representations of the underlying system. *ii)* Perform a fast Monte Carlo search of the DNO functional space for extremes. *iii)* Compute statistics over a Monte Carlo ensemble and select new input functions that both explore and exploit the space for extremes. *iv)* Evaluate proposed inputs on the underlying experiment or simulation, record outputs, and pass to *i)*. Repeat *i)-iv)* until statistics are converged or resources are depleted. *End)* with an optimally trained DNO that supports prediction of extreme events.

Neural Operators [21], Convolutional NNs, Recurrent NNs, Long-Short Term Memory, among others. In fact, this work did not explicitly investigate the full value of DeepONet for operator learning for BED. Instead, only the standard feed-forward branch NN was used. Using the complete operator machinery only requires adjusting the parameters related to the operator and trunk. Here, we kept these parameters constant. Finally, while we focus on Bayesian experimental design, slight adjustments to the choice of acquisition function allow for Bayesian Optimization tasks [22] or for approaching other metrics of interest (e.g. mean squared error).

In conclusion, we believe we have demonstrated an equation-agnostic framework that: (1) efficiently discovers extreme events, (2) is computationally tractable, and (3) presents a straightforward implementation on any arbitrary input-output system. This creates a unique opportunity for experimentalists and computationalists alike to investigate their systems for extreme behavior, whether that behavior originates from societal or physical systems or has beneficial or catastrophic consequences.

4 Methods

Our approach to Bayesian experimental design is detailed in figure 5 and consists of two critical components, *data selection criteria* and the *surrogate model.* Algorithm 1 formalizes the iterative steps taken in figure 5 for efficiently training a surrogate model with minimal data selection.

4.1 Surrogate Models

We test two surrogate models in the BED framework, GPs as the benchmark case, and DNOs (specifically DeepONet) for greater scaling and generalization performance. While both GPs and DNOs provide the same role in the framework, their implementation is fundamentally different. Whereas GPs are used to map random parameter inputs $\boldsymbol{\theta}$ to an output $f(\boldsymbol{\theta})$, we use the DNOs to map an input function $u = \boldsymbol{\theta}\boldsymbol{\Phi}$ to an

Algorithm 1 Sequential search for active learning/training of DNOs and GPs .

```
1: Input: Number of iterations: $n_{iter}$
2: Initialize: Train GP/DNO on initial dataset of input-output pairs
   GP: $\mathcal{D}_0 = \{\boldsymbol{\theta}_i, y_i = f(\boldsymbol{\theta}_i)\}_{i=1}^{n_{\text{init}}}$    DNO: $\mathcal{D}_0 = \{\boldsymbol{\theta}_i, y_i = G(\boldsymbol{\theta}_{u,i}\boldsymbol{\Phi}(x_1, ...x_m), \boldsymbol{\theta}_{z,i})\}_{i=1}^{n_{\text{init}}}$
3: for $n = 1$ to $n_{iter}$ do
4:     Select next point $\boldsymbol{\theta}_n$ by minimizing* acquisition function $a(\boldsymbol{\theta})$:
         *Minimization using Monte Carlo, See Section 4.3 and Appendix F
         $$\boldsymbol{\theta}_n = \arg\min_{\boldsymbol{\theta}\in\mathcal{X}} \; a(\boldsymbol{\theta}; y), \mathcal{D}_{n-1})$$
5:     Evaluate objective function at $\boldsymbol{x}_n$ and record $y_n$
6:     Augment dataset: $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \boldsymbol{\theta}_n, y_n$
7:     Retrain GP/DNO.
8: end for
9: return Final GP/DNO Model
```

output function $G(u = \boldsymbol{\theta}\boldsymbol{\Phi})$. In both cases, $\boldsymbol{\theta}$ is the only independent variable and is maintained to provide a means of searching for extremes in the parameter space. However, the proposed regression task for DNOs ensures that the they perform their mapping in functional, or physical, space, rather than the parameter space. This distinction is foundational for our approach and success with DNOs for BED.

We detail both GPs and DeepONet in detail in Appendices C and D, respectively, and only discuss our approach for quantifying the predictive variance, $\sigma^2(\boldsymbol{\theta})$, for DNOs using ensembles.

4.1.1 Ensemble of Neural Networks for Uncertainty Quantification

Although neural network architectures are attractive for approximating nonlinear regression tasks, their complexity rids them of analytical expressions. This does not allow for a traditional Bayesian treatment of uncertainty in the underlying surrogate model – a key property present for GP regression (see equation 22). Knowledge of the uncertainty of a surrogate model allows one to target model deficiencies as seen in the parameter space. This means that choosing an appropriate method for quantifying the uncertainty is a crucial and key component to active learning or BED. There are several techniques for quantifying uncertainty in neural networks and we provide a brief description of these in Appendix D.1 and focus only on ensemble methods.

Ensemble approaches have been used extensively throughout the literature [23, 24] and despite their improved results for identifying the underlying tasks at hand [25], their utility for quantifying uncertainty in a model remains a topic of debate. There are several approaches for creating ensembles. These include random weight initialization[24], different network architectures (including activation functions), data shuffling, data augmentation, bagging, bootstrapping, and snapshot ensembles [26, 27, 28] among others. Here we employ random weight initialization, a technique found to perform similarly or better than BNN approaches (Monte Carlo Dropout and Probablistic Backpropagation) for evaluation accuracy and out-of-distribution detection for both classification and regression tasks [24]. As stated earlier, much of the field is skeptical of the generality of ensembles to provide rigorous uncertainty estimates. However, recent studies, such as [14] and specifically [15], have argued that DNN and DNO ensembles provide reasonable, if not superior, approximations of the posterior. Finally, the straightforward implementation of the randomly initialized weights motivates our choice, as it makes the adoption of these techniques far more probable.

We train N randomly weight-initialized DNO models, each denoted as $\tilde{G}_n$, that find the associated solution field y for inputs u and z. This allows us to then determine the point-wise variance of the models as

$$\sigma^2(u, z) = \frac{1}{(N-1)} \sum_{n=1}^{N} (\tilde{G}_n(u)(z) - \overline{\tilde{G}(u)(z)})^2, \tag{3}$$

where $\overline{\tilde{G}(u)(z)}$ is the mean solution of the model ensemble. Finally, we must adjust the above representation to match the description for BED. In the case of traditional BED and GPs, the input parameters, $\boldsymbol{\theta}$, represent the union of two set of parameters, $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_z$. The parameters $\boldsymbol{\theta}_u$ typically represent coefficients to a set of functions that represent a decomposition of a random function $u = \boldsymbol{\theta}_u \boldsymbol{\Phi}(x_1, ...x_m)$, while $\boldsymbol{\theta}_z = z$ represent non-functional parameters of the operator. Thus, the DNO description for UQ may be recast as:

$$\sigma^2(\boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}_u \cup \boldsymbol{\theta}_z) = \frac{1}{(N-1)} \sum_{n=1}^{N} (\tilde{G}_n(\boldsymbol{\theta}_u \boldsymbol{\Phi}(x_1, ...x_m))(\boldsymbol{\theta}_z) - \overline{\tilde{G}(\boldsymbol{\theta}_u \boldsymbol{\Phi}(x_1, ...x_m))(\boldsymbol{\theta}_z)})^2. \tag{4}$$

4.2 Data Selection: Acquisition Functions

The acquisition function is the key component of the sequential search algorithm, as it guides algorithm 1 in exploring the input/parameter space and determines points at which the objective function is to be queried. Because of the lack of a closed analytical form of DNOs, we only consider two acquisition functions used previously with GPs on several test cases in [16]. The two functions we are interested in are the commonly used uncertainty sampling and the output-weighted uncertainty sampling proposed by [16]. The novelty here is that we apply them explicitly to DNOs (DeepONet) and present the advantages of DNOs compared to GPs as we apply them to a complex and high-dimensional problem.

4.2.1 Uncertainty Sampling

Uncertainty sampling (US) is one of the most broadly used active sampling techniques and identifies the point where the predictive variance is the greatest,

$$a_{US}(\boldsymbol{\theta}) = \sigma^2(\boldsymbol{\theta}). \tag{5}$$

Uncertainty sampling, also known as the active-learning-MacKay (ALC) algorithm [29], imposes an sequential search that evenly distributes uncertainty over the input space as it gains data. The popularity of US is due to three qualities: ease of implementation, inexpensive evaluation (for small datasets with GPs), and analytic gradients, the last of which permits the use of gradient-based optimizers.

4.2.2 Likelihood-Weighted Acquisition Functions

There are several "extreme event" likelihood-weighted acquisition functions that could be explored, as proposed by [5], but we elect to only test the likelihood-weighted uncertainty sampling (US-LW) acquisition because of its simplicity in implementation. For US-LW, we augment the US sampling acquisition function with the previously described danger scores to give

$$a_{US-LW}(\boldsymbol{\theta}) = w(\boldsymbol{\theta})\sigma^2(\boldsymbol{\theta}), \tag{6}$$

such that both *highly uncertain* and *highly dangerous* regions are sampled.

To compute $w(\boldsymbol{\theta})$, we note that the approximated output PDF, $p_\mu(\mu(\boldsymbol{\theta}))$ is approximated via a kernel density estimator with $n = 10^6$ test points. For the DNO cases, we chose to compute this with only the first ensemble member, $\mu = \tilde{G}_1$, to reduce computational costs. Similar to the ensemble results for $N = 2$, using only one ensemble member is akin to using Thompson sampling and performs without reduction in performance.

4.2.3 Danger Maps and Log-PDF Error Metrics

To test the ability of the DNO and GP Bayesian-inspired sequential algorithms to quantify extremes, we define a "Danger Map" for the pandemic scenarios and rogue waves. The rogue wave danger map is defined as,

$$f(\boldsymbol{\theta}) = ||Re(u(x, t = T; \boldsymbol{\theta}))||_\infty, \tag{7}$$

where $T = 20$, while the pandemic danger map is,

$$f(\boldsymbol{\theta}) = I(t = T; \boldsymbol{\theta}), \tag{8}$$

where $T = 75$ days. For each case, we then select a random set, chosen through Latin hypercube sampling, of 10^5 test points, $\mathbf{\Theta} \in \mathbb{R}^{d\times 10^5}$ and use a kernel density estimator to compute the true PDF $p_f(y)$, computed as

$$p_f(y) = \int_{\boldsymbol{\theta}} p_f(y|\mathbf{\Theta})p_{\boldsymbol{\theta}}(\mathbf{\Theta})|\mathrm{d}\boldsymbol{\theta}, \tag{9}$$

and the model's prediction $p_{\mu_n}(y)$ for the same test points, $\mathbf{\Theta}$. To determine whether the testing data appropriately identifies extremes, we compute the log-PDF error

$$e(n) = \int |\log_{10} p_{\mu_n}(y) - \log_{10} p_f(y)|\mathrm{d}y. \tag{10}$$

4.3 Monte Carlo Minimization of Acquisition Functions

In our experiments, we consistently observe that acquisition points found through minimizers using gradient descent are not globally optimal. Instead, Monte Carlo evaluation of the DNOs and GPs consistently find superior optima. This is chiefly because of the non-convexity of the acquisition function. We may recall the highly non-convex behavior of the 2D acquisition fields in figure 2 *b*), even with as little as ≈ 5 samples. This non-convex nature emits many local minima that requires many initial search points to provide confidence that the chosen optima is nearly global. As the minimizer progresses for each iteration, it must call the DNO or GP, whereas a Monte Carlo approach may efficiently evaluate all points in one vector operation. This means that for the same computation time as the minimizers, Monte Carlo sampling may evaluate a substantially larger distribution of query points and return acquisition points with superior scores than that of the optimizer. In Appendix F we show that acquisition scores found via Monte Carlo at $20D$ consistently outperform minimizers for similar computation times.

4.4 Experiment Batching

As systems become more complex, additional experiments/data are required to reduce errors for higher dimensional cases, as observed in figure 3. Considering many experiments can be conducted in parallel, we ask whether choosing multiple local minima of the acquisition function presents marginally reduced performance than a purely sequential search. This is especially critical for situations where experimental time is more costly than additional setups (e.g. protein or genetic design).

The purpose of batching is to find *multiple regions of local optima* of the acquisition function, rather than finding *several optima in the same region*. To impose this idea, we create a constraint that no acquisition point may reside closer than a distance $r_{\min}$ to each other. We define $r_{\min}$ as a fraction of the maximum euclidean distance of the space being sampled,

$$r_{\min} = r_l \left(\sum_{d=1}^{D} (\theta_{d,+} - \theta_{d,-})^2 \right)^2, \tag{11}$$

where $\theta_{d,+}$ and $\theta_{d,-}$ are the maximum and minimum domain bounds of each parameter dimension d and r_l is the user defined percentage. In this work, we chose a static $r_l = 0.025$, but dynamic values based on the packing of the parameter space would be an intriguing direction for increasing the efficacy of this approach. Imposing this constraint requires an iterative processing of the acquisition scores, detailed in Algorithm 2. For the batching applied in this study, each case uses a Monte Carlo querying of $n_q = 10^6$ points.

References

[1] Creating a disaster resilient america: Grand challenges in science and technology. *National Academies Press*, 2005.

[2] O. E. Hansteen, H. P. Jostad, and T. I. Tjelta. Observed platform response to a monster wave. In *Field Measurements in Geomechanics*, pages 73–86, 2003.

Algorithm 2 Sequential selection of points for batch sampling

```
1: Evaluate acquisition function for $n_q$ query points.
2: for Acquisition points smaller than batch size $n_a < n_b$
3:     Choose the minimum score, $\min(a) = a(\boldsymbol{\theta}_c)$, from $n_q$ points.
4:     Augment $\boldsymbol{\Theta}_a$ with chosen point, $\boldsymbol{\theta}_c$.
5:     Compute the distances, $r$, between the chosen point and the remaining query points.
6:     Eliminate all points from $n_q$ where $r < r_{\min}$.
7: end for
8: return Points for the next experiment: $\boldsymbol{\Theta}_a$
```

[3] J. Gemmrich and L. Cicon. Generation mechanism and prediction of an observed extreme rogue wave. *Scientific Reports*, 12(1):1–10, 2022.

[4] T. P. Sapsis. Statistics of extreme events in fluid flows and waves. *Annual Review of Fluid Mechanics*, 53:85–111, 2021.

[5] A. Blanchard and T. P. Sapsis. Bayesian optimization with output-weighted optimal sampling. *Journal of Computational Physics*, 425:109901, 2021.

[6] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[7] H. Kahn and A. W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

[8] M. Shinozuka. Basic analysis of structural safety. *Journal of Structural Engineering*, 109(3):721–740, 1983.

[9] G. Dematteis, T. Grafke, and E. Vanden-Eijnden. Extreme event quantification in dynamical systems with random components. *SIAM/ASA Journal on Uncertainty Quantification*, 7(3):1029–1059, 2019.

[10] F. Uribe, I. Papaioannou, Y. M. Marzouk, and D. Straub. Cross-entropy-based importance sampling with failure-informed dimension reduction for rare event simulation. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):818–847, 2021.

[11] S. Wahal and G. Biros. Bimc: The Bayesian Inverse Monte Carlo method for goal-oriented uncertainty quantification. Part I. *arXiv preprint arXiv:1911.00619*, 2019.

[12] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 18:1257, 2006.

[13] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574. PMLR, 2009.

[14] E. Pickering and T. P. Sapsis. Structure and distribution metric for quantifying the quality of uncertainty: Assessing gaussian processes, deep neural nets, and deep neural operators for regression. *arXiv preprint arXiv:2203.04515*, 2022.

[15] A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

[16] A. Blanchard and T. P. Sapsis. Output-weighted optimal sampling for Bayesian experimental design and uncertainty quantification. *arXiv e-prints*, pages arXiv–2006, 2020.

[17] M. A. Mohamad and T. P. Sapsis. Sequential sampling strategy for extreme event statistics in nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 115(44):11138–11143, 2018.

[18] T. P. Sapsis. Output-weighted optimal sampling for Bayesian regression and rare event statistics using few samples. *Proceedings of the Royal Society A*, 476(2234):20190834, 2020.

[19] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. *Advances in Neural Information Processing Systems*, 24, 2011.

[20] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[21] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[22] Y. Yang, A. Blanchard, T. P. Sapsis, and P. Perdikaris. Output-weighted sampling for multi-armed bandits with extreme payoffs. *arXiv preprint arXiv:2102.10085*, 2021.

[23] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[24] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

[25] F. K. Gustafsson, M. Danelljan, and T. B. Schon. Evaluating scalable Bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020.

[26] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[27] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[28] L. N. Smith. No more pesky learning rate guessing games. *CoRR, abs/1506.01186*, 5, 2015.

[29] R. B. Gramacy and H. K. H. Lee. Adaptive design and analysis of supercomputer experiments. *Technometrics*, 51(2):130–145, 2009.

[30] A. J. Majda, D. W. McLaughlin, and E. G. Tabak. A one-dimensional model for dispersive wave turbulence. *Journal of Nonlinear Science*, 7(1):9–44, 1997.

[31] D. Cai, A. J. Majda, D. W. McLaughlin, and E. G. Tabak. Spectral bifurcations in dispersive wave turbulence. *Proceedings of the National Academy of Sciences*, 96(25):14216–14221, 1999.

[32] V. E. Zakharov, P. Guyenne, A. N. Pushkarev, and F. Dias. Wave turbulence in one-dimensional models. *Physica D: Nonlinear Phenomena*, 152:573–619, 2001.

[33] V. E. Zakharov, F. Dias, and A. Pushkarev. One-dimensional wave turbulence. *Physics Reports*, 398(1):1–65, 2004.

[34] A. Pushkarev and V. E. Zakharov. Quasibreathers in the MMT model. *Physica D: Nonlinear Phenomena*, 248:55–61, 2013.

[35] W. Cousins and T. P. Sapsis. Quantification and prediction of extreme events in a one-dimensional nonlinear dispersive wave model. *Physica D: Nonlinear Phenomena*, 280:48–58, 2014.

[36] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[37] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[38] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

[39] L. Lu, P. Jin, and G. E. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[40] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

[41] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. Em. Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *arXiv preprint arXiv:2201.07766*, 2022.

[42] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. *arXiv preprint arXiv:1806.01768*, 2018.

[43] A. Malinin and M. Gales. Predictive uncertainty estimation via prior networks. *arXiv preprint arXiv:1802.10501*, 2018.

[44] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[45] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

[46] Y. Gal, R. Islam, and Z. Ghahramani. Deep Bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

A SIR Pandemic Model

We implement a simple Susceptible, Infected, Recovered (SIR) model,

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta(t)IS + \delta R \tag{12}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta(t)IS - \gamma I \tag{13}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I - \delta R, \tag{14}$$

where δ is the rate of immunity loss, γ is the recovery rate, and $\beta(t)$ is a stochastic infection rate. Here we take $\delta = 0$ and $\gamma = 0.1$ and the stochastic infection rate, $\beta(t)$, is defined as

$$\beta(t) = \beta_0(\theta\mathbf{\Phi}(t) + \phi_0), \tag{15}$$

where $\Phi(t)$ is found via a Karhunen-Loeve expansion of a radial basis kernel with $\sigma_\beta^2 = 0.1$ and length scale $\ell_\beta = 0.1$ and $\phi_0 = 2.15$, to ensure all infection rates are non-negative. Initial conditions for the model are $I_0 = 50$ with total population $P = 10^8$, and a step size of 0.2 days is used.

B Dissipative Nonlinear Schrödinger Equation: Majda, McLaughlin, and Tabak Model

The Majda, McLaughlin, and Tabak [30] model is a dissipative version of the nonlinear Schrödinger (NLS) equation used for studying 1D wave turbulence. It is a one-dimensional, dispersive nonlinear prototype model with intermittent events described by

$$iu_t = |\partial_x|^\alpha u + \lambda |\partial_x|^{-\beta/4} \left(\left||\partial_x|^{-\beta/4} u\right|^2 |\partial_x|^{-\beta/4} u \right) + iDu, \tag{16}$$

where u is a complex scalar, exponents α and β are chosen model parameters, and D is a selective Laplacian (described further below). This model gives rise to four-wave resonant interactions that, especially when coupled with large scale forcing and small scale damping, produces a family of spectra revealing both direct and inverse cascades [30, 31]. A realization of the MMT model is shown in figure 6 that demonstrates these complex dynamical properties. Not only does this model provide a rich dynamical response, but also presents a unique utility as a physical model for extreme ocean waves, or rogue waves [32, 33, 34]. Hence, its study is an ideal test bed for both examining the numerical difficulties of predictive models and uncovering insights to physical, real-world applications.

For both ease of computation and for discussion of the terms in the MMT model, we transform the equation into wavenumber space. The pseudodifferential operator $|\partial_x|^\alpha$, via the Fourier transform in space becomes: $\widehat{|\partial_x|^\alpha u}(k) = |k|^\alpha \widehat{u}(k)$ where k is the wavenumber in x. This formulation may be similarly defined on a periodic domain. We choose $\alpha = 1/2$ and $\beta = 0$ as done in [35], reducing equation (16) to

$$\widehat{u}(k)_t = -i|k|^{1/2}\widehat{u}(k) - i\lambda|\widehat{u}(k)|^2\widehat{u}(k) + \widehat{Du}(k) + f(k), \tag{17}$$

where $f(k)$ is a forcing and $\widehat{Du}(k)$ is a selective Laplacian of the form:

$$\widehat{Du}(k) = \begin{cases} -\left(|k| - k^*\right)^2 \hat{u}(k) & |k| > k^* \\ 0 & |k| \leq k^* \end{cases} \tag{18}$$

where k^* presents the lower bound of wavenumbers subject to dissipation. For the model considered in this study we choose $\lambda = -0.5$, $k^* = 20$, $f(k) = 0$, $dt = 0.001$, and a grid that is periodic between 0-1 with $N_x = 512$ grid points. To propose a stochastic and complex initial condition, $u(x, t = 0)$, we take the complex-valued kernel

$$k(x, x') = \sigma_u^2 e^{i(x-x')} e^{-\frac{(x-x')^2}{\ell_u}}, \tag{19}$$

with $\sigma_u^2 = 1$ and $\ell_u = 0.35$. We then parametrize the stochastic initial conditions by a finite number of random variables, $\boldsymbol{\theta}$, using the Karhunen-Loeve expansion of the kernel's correlation matrix,

$$u(x, t = 0) \approx \boldsymbol{\theta}\boldsymbol{\Phi}(x), \qquad \forall \qquad x \in [0, 1) \tag{20}$$

where $\boldsymbol{\theta} \in \mathbb{C}^m$ is a vector of complex coefficients and both the real and imaginary components of each coefficient are normally distributed with zero mean and diagonal covariance matrix Λ, and $\{\boldsymbol{\Lambda}, \boldsymbol{\Phi}(x)\}$ contains the first m eigenpairs of the correlation matrix. This gives the dimension of the parameter space as $2m$ due to the complex nature of the coefficients. For all cases, the random variable θ_i is restricted to a domain ranging from -6 to 6, in that 6 standard deviations in each direction from the mean.

C Gaussian Process Regression

For low-dimensional problems, Gaussian process (GP) regression [36] is the "gold standard" for Bayesian design. There are several attractive qualities of GPs. They are agnostic to the details of a black-box process (just like neural networks described next) and they clearly quantify both the uncertainty in the model and the uncertainty associated with noise. GPs are also relatively easy to implement and cheap to train.

A Gaussian process $\bar{f}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a random variable, is completely specified by its mean function $m(\boldsymbol{\theta})$ and covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$. For a dataset $\mathcal{D}$ of input-output pairs $(\{\boldsymbol{\Theta}, \mathbf{y}\})$ and a Gaussian process with constant mean $m_{\mathbf{0}}$, the random process $\bar{f}(\boldsymbol{\theta})$ conditioned on $\mathcal{D}$ follows a normal distribution with posterior mean and variance

$$\mu(\boldsymbol{\theta}) = m_0 + k(\boldsymbol{\theta}, \boldsymbol{\Theta})\mathbf{K}^{-1}(\mathbf{y} - m_0) \tag{21}$$

$$\sigma^2(\boldsymbol{\theta}) = k(\boldsymbol{\theta}, \boldsymbol{\theta}) - k(\boldsymbol{\theta}, \boldsymbol{\Theta})\mathbf{K}^{-1}k(\boldsymbol{\Theta}, \boldsymbol{\theta}) \tag{22}$$

respectively, where $\mathbf{K} = k(\boldsymbol{\Theta}, \boldsymbol{\Theta}) + \sigma_\epsilon^2\mathbf{I}$. Equation (21) can predict the value of the surrogate model at any point $\boldsymbol{\theta}$, and (22) to quantify uncertainty in prediction at that point [36]. Here, we chose the radial-basis-function (RBF) kernel with automatic relevance determination (ARD),

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma_f^2 \exp\left[-(\boldsymbol{\theta} - \boldsymbol{\theta}')^\top \mathbf{L}^{-1}(\boldsymbol{\theta} - \boldsymbol{\theta}')/2\right], \tag{23}$$

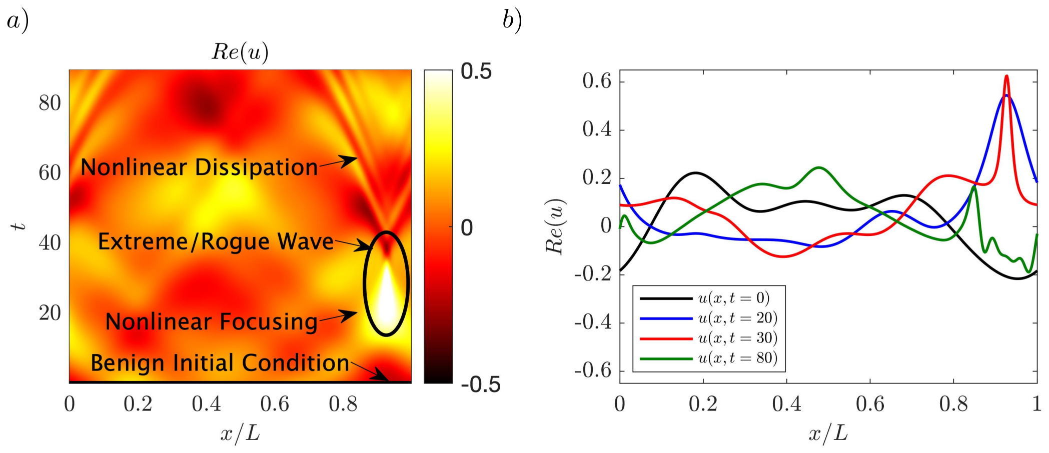

Figure 6: **The NLS emits many nonlinear phenomena in finite time, from focusing to dissipation.** *a*) A realization of the NLS model ($\alpha = 1/2, \beta = 0, \lambda = -0.5$) we discover with DNO+BED, where a benign initial condition leads to an extreme, rogue wave. *b*) Wave height of selected times from *a*).

where $\boldsymbol{L}$ is a diagonal matrix containing the lengthscales for each dimension and the GP hyperparameters appearing in the covariance function (σ_f^2 and $\boldsymbol{L}$ in (22) are trained by maximum likelihood estimation).

One setback from the above expression is the inference step in GP regression, where each iteration requires the inversion of the matrix $\mathbf{K}$. Typically performed by Cholesky decomposition, the inversion cost scales as $O(n^3)$, with n being the number of observations [36, 37]. This means that as problems grow to infinite dimensions, inevitably requiring larger datasets, GPs become prohibitively costly. We will show here that as the need for data increases, GPs become much more computationally intensive than our next surrogate model, neural networks.

D Deep Neural Operators and DeepONet

Unlike GPs, Deep Neural Operators or Deep Neural Networks, do not suffer from data-scaling challenges and are our primary model class for consideration in this manuscript. Deep Neural Networks, when cast as neural operators[38], are specifically well-suited for characterizing infinite dimensional systems, as they may map functional inputs to functional outputs. Although our work is general for any neural network approach, we leverage the architecture proposed by [6] for approximating nonlinear operators: *DeepONet*.

DeepONet seeks approximations of nonlinear operators by constructing two deep neural networks, one representing the input function at a fixed number of sensors and another for encoding the "locations" of evaluation of the output function. The first neural network, termed the "branch", takes input functions, u, observed at discrete sensors, $x_i, i = 1...m$. These input functions can take on several representations, such as initial conditions (i.e. u_0) or forcing functions. The second neural network, termed the "trunk", should be seen as an encoder for inherent qualities of the operator, denoted as z (referred to as y in [6], but changed here for typical active sampling notation). For example, a variable coefficient or exponent in the true nonlinear operator, G, alternatively, and the usual use case for DeepONet, the trunk variable can refer to the evaluation of the operation to an arbitrary point in time and/or space. Together, these networks seek to approximate the nonlinear operation upon u and z as $G(u)(z) = y$, where y will denote the scalar output from the u, z input pair.

Given a set of input-output pairs, $\{[\mathbf{u}, \mathbf{z}], G(\mathbf{u})(\mathbf{z})\}$, DeepONet seeks to minimize the difference between the true operator, $G(u)(z)$, and the dot product between two neural networks $\mathbf{g}(u)$ and $\mathbf{f}(z)$. These network's level of expressivity is governed by the number of neurons (n), the number of layers (l_b, l_t for branch and trunk, respectively), and activation functions. Under sufficient training DeepONet can meet any arbitrary error, ϵ, as

$$| G(u)(z) - \langle \underbrace{\mathbf{g}\left(u\left(x_1\right), u\left(x_2\right), \cdots, u\left(x_m\right)\right)}_{\text{branch}}, \underbrace{\mathbf{f}(z)}_{\text{trunk}} \rangle | < \epsilon. \tag{24}$$

Thus, DeepONet can, to an arbitrarily small precision, approximate an infinite dimensional nonlinear operator $G(u)(z)$. However, the functions $\mathbf{g}$ and $\mathbf{f}$ are typically trained under the assumption of plentiful data. We are interested in how DeepONet can be optimally trained, in that with the least amount of data, for discovering and quantifying extreme events.

DeepONet is attractive for these tasks as neural nets generalize well and are incredibly fast to evaluate (compared to their experimental/simulation counterparts) for arbitrarily chosen points. However, there is little work on how one optimally selects the best samples to train a neural net. Previously, the approach taken has included creating an appropriate basis for specific operators that are used to train DeepONet [39]. For the problems we are interested (i.e. rare and extreme events), this appropriate basis is unknown. Therefore, we envision DeepONet will provide a flexible model to learn seen data, and then provide early predictions of where danger and uncertainty lie in the input/output space through DeepONet's parameterization, that we may then query the underlying system to best inform DeepONet.

D.1 Approaches for quantifying uncertainty in Deep Neural Networks

There are several techniques for quantifying uncertainty in neural networks and three categorizations of these techniques we wish to mention: single deterministic networks, Bayesian neural network (BNN) inference approaches, and ensemble methods (see [40] and [41] for comprehensive reviews of these methods for uncertainty quantification). Unfortunately, a complex array of advantages and disadvantages of each approach provides no clear favorite. Single deterministic methods [42, 43] use one forward pass of a deterministic network to learn both the mean and variance of a labeled output. Using only one model leads to cheap training and evaluation. However, this single opinion of the underlying system results in substantial sensitivity, an unattractive quality for regression problems quantifying physical instabilities/extreme events. Bayesian neural networks [44, 45, 46, 15] encompass a broad variety of stochastic DNNs that combine Bayesian inference theory with the expressiveness, scalability, and predictive performance enjoyed by deep neural networks. Although the supporting theory behind Bayesian methods, as well as empirical results, imparts faith that such models will lead to the greatest chance of success, they do come with a significant disadvantage, complexity. BNNs are significantly more complex than standard neural networks and can be exhausting to train, making them difficult to implement. From an academic viewpoint, this can be overcome. However, as a study concerned with translating DNNs and active learning to practical engineering systems that solve real-world problems, we opt for using the straight-forward ensemble approach. An approach that seats itself between the single deterministic model and the infinite model representations of BNNs.

E DeepONet Setup

Application of the NLS and SIR models to the DeepONet architecture requires a well-posed distinction between the functions and parameters that belong in the branch and trunk networks. Figure 7 provides a visual of the function/parameter delineations and the architecture for each modeling task.

In this work, we provide all input initial conditions as functions to the branch network. Because of the requirement that all inputs to DeepONet take on real values, the NLS setup requires that we split the input function, u, into its real (u_r) and imaginary (u_i) components. It is critical to keep both components, as each contains unique and coupled information that is propagated in the NLS operation (this is not necessary for the real-valued SIR model). We then stack these components as a vector by x position/sensor with the imaginary component directly following the real component at each x position. Technically, the ordering of the inputs does not matter, due to the linear nature of the first layer, but the ordering of the inputs must remain consistent. For increased speed, we also reduce the sensor number of x_m sensors from the 512 used for the direct NLS calculation to 128 points for DeepONet (this corresponds to 256 total inputs values because of the real and imaginary components) and 125 points for the SIR infection rate. We also give the length scale and variance as input parameters, as they define the kernel that emits the Karhunen-Loeve expansion functions. The input functions directly recognize adjustments to these parameters.

The trunk contains parameters that are intrinsic to the NLS and SIR operations. For NLS, these comprise both time and space (t, x) and the chosen constants $\alpha, \beta, \lambda, k^*$. Regarding BED, we may assign each of these parameters to an appropriate prior distribution, such as uniform for space and time. In this study, we

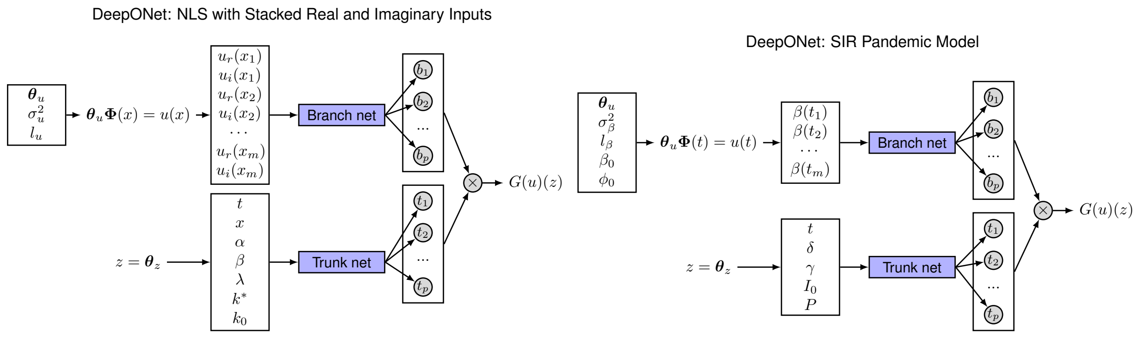

Figure 7: **The application of the NLS and SIR operators to DeepONet.**

Case	Neurons	Branch Layers	Ensemble Size	Experiments
$2-6D$	200	5	10	10
$8D$ Batching	200	5	10	10
$8D$ Ensemble, $n \leq 2500$	200	5	2,4,8,16	10
$8D$ Ensemble, $n > 2500$	200	6	2,4,8,16	10
$20D$, $n \leq 2500$	200	5	2	25
$20D$, $n > 2500$	200	6	2	25

Table 1: **Hyperparameters used for all cases.** All trunk layers = 1 due to their redundancy.

choose all values to remain constant and although the trunk is straightforward to implement; it is simply a redundant network here.

We provide the hyperparameters and other quantities used for the various NLS cases performed throughout the manuscript in table 1. In addition to the hyperparameters, we use the ReLu activation function, a learning rate of $l_r = 0.001$, and 1000 epochs for each training procedure. For the SIR search, the same hyperparameters are used with 6 branch layers, 1 redundant trunk layer, and 200 neurons.

F Monte Carlo Minimization at $20D$

Here we show that acquisition points found through Monte Carlo (MC) searches are consistently superior than those selected by minimizers using gradient descent algorithms (when reasonably similar computation times are considered). To demonstrate this behavior, we test four methods for finding 50 batched acquisition points for the $20D$ NLS case at 500, 2500, and 5000 training samples, $N = 2$ ensemble members, and over 25 independent experiments. The four approaches are listed below, with each case choosing the 50 best points with algorithm 2:

1. L-BFGS-B algorithm, implemented within the `scipy Python` package, 10^2 random LHS initial points.
2. MC with 10^5 random uniform initial points (sampled from a uniform distribution for speed)
3. MC with 10^5, best 100 points (algorithm 2) passed to L-BFGS-B.
4. MC with 10^6 random uniform initial points.

Table 2 and table 3 provide the mean computation times and scores for all cases, respectively. The mean computation times ($\pm$ one standard deviation) are clearly faster for MC methods, by 5-fold, compared to any approach with the L-BFGS-B algorithms. We also note that only the first ensemble member is used for computing and querying p_μ for speed, as well as likely better performance.

Table 3 provides the mean scores (over all chosen batch points and experiments, $n = 1250$), as well as the mean best and worst point for each experiment ($n = 25$). It is critical to note that at these high dimensions the difference between acquisition scores can be *hundreds* of orders of magnitude different (this is a direct result of weights define by the ratio of two PDFs computed over 20 dimensions). To navigate this

Samples	L-BFGS-B	MC: 10^5	MC+L-BFGS-B	MC: 10^6
500	122 ± 7	3.6 ± 0.1	123 ± 5	$\mathbf{27 \pm 1}$
2500	126 ± 6	4.0 ± 0.2	125 ± 5	$\mathbf{26 \pm 2}$
5000	127 ± 8	3.7 ± 0.1	135 ± 7	$\mathbf{27 \pm 3}$

Table 2: **MC is faster than built-in optimizers.** Mean compute times ($n = 25$) for batch minimization ($n_b = 50$) of the the $20D$ US-LW acquisition function with $\pm$ one standard deviation. Best values are bold.

Samples	L-BFGS-B	MC: 10^5	MC+L-BFGS-B	MC: 10^6
Mean ($n = 1250$)				
500	110 ± 18	70 ± 5	66 ± 6	$\mathbf{60 \pm 4}$
2500	104 ± 16	67 ± 5	63 ± 5	$\mathbf{56 \pm 4}$
5000	110 ± 17	76 ± 4	67 ± 8	$\mathbf{64 \pm 4}$
Min $n = 25$				
500	$\mathbf{42} \pm 32$	57 ± 6	53 ± 16	$52 \pm \mathbf{5}$
2500	$\mathbf{44} \pm 32$	$53 \pm \mathbf{4}$	48 ± 10	47 ± 6
5000	53 ± 21	60 ± 5	$\mathbf{48} \pm 11$	$54 \pm \mathbf{3}$
Max $n = 25$				
500	126 ± 7	73 ± 3	70 ± 3	$\mathbf{63 \pm 3}$
2500	117 ± 6	70 ± 3	65 ± 3	$\mathbf{60 \pm 3}$
5000	125 ± 5	80 ± 2	72 ± 2	$\mathbf{67 \pm 2}$

Table 3: **MC optimization provides consistently superior acquisition scores over built-in optimizers.** Top rows: The mean and standard deviation of all 1250 chosen acquisition points for the four methods on DNOs trained with 500, 2500, and 5000 samples. Middle rows: The mean and standard deviation of the best acquisition points for the 25 experiments. Bottom rows: The mean and standard deviation of the worst chosen acquisition points (i.e. the least optimal point of the batch). All values are $-\log_{10}(a(\boldsymbol{\theta}))$. Best values are bold.

computational challenge, we take the $\log_{10}$ of the scores and apply a negative sign such that lower scores are optimal (i.e. $-\log_{10}(a(\boldsymbol{\theta}))$). Considering the mean of all chosen points, we see that a naive use of the L-BFGS-B minimizer finds scores that are ≈ 50 orders of magnitude worse than the 10^6 MC approach, regardless of the training complexity of the model. The L-BFGS-B minimizer only provides a marginal advantage for finding the best point in any random instance, however, the standard deviation is 30 orders of magnitude and suggests the approach is terribly inconsistent. Juxtaposing the min and max results, we see that the MC method only varies by approximately 10 orders of magnitude compared to the 80 orders of the L-BFGS-B.

Overall, the MC method provides greater consistency in finding several attractive acquisition points, while also enjoying ease of implementation and better computational efficiency. Of course, if given enough computational resources and time, the L-BFGS-B method permits pinpointing exceptional acquisition points. Approaches combining both MC and L-BFGS-B, where the MC points are further optimized (as also implemented here) or performed from more LHS points, can either refine or enrich the set of acquisition points.

We believe the results found here are chiefly because of the non-convexity of the acquisition function. We may recall the highly non-convex behavior of the 2D acquisition fields in figure 2b), even with as little as ≈ 10 samples. This non-convex nature emits many local minima that requires many initial points to provide confidence that the chosen optima is nearly global. This makes the task for built in minimizers extremely difficult, especially for high dimensional problems, and lead to minimizations that become easily fooled or stuck. This difficulty is so challenging that the sparse MC sampling of only 10^6 points, an extremely sparse sampling of a $20D$ space, easily outperforms the minimizers.