Computational Materials Science 201 (2022) 110883

Contents lists available at ScienceDirect

Computational Materials Science

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

Uncertainty bounds for multivariate machine learning predictions on high-strain brittle fracture

Cristina Garcia-Cardona [a,*], M. Giselle Fernández-Godino [b], Daniel O'Malley [c], Tanmoy Bhattacharya [d]

[a] MS B256, Information Sciences Group (CCS-3), Los Alamos National Laboratory, Los Alamos, NM 87545, United States of America
[b] L-103, Atmospheric Science Research & Applications Group, AEED, Lawrence Livermore National Laboratory, 7000 East Ave, Livermore, CA 94550, United States of America
[c] MS D446, Computational Earth Sciences Group (EES-16), Los Alamos National Laboratory, Los Alamos, NM 87545, United States of America
[d] MS B285, Nuclear and Particle Physics, Astrophysics & Cosmology Group (T-2), Los Alamos National Laboratory, Los Alamos, NM 87545, United States of America

ARTICLE INFO

Keywords:
Machine learning
Crack statistics
Uncertainty quantification
Heteroscedastic approach

ABSTRACT

Simulation of the crack network evolution on high strain rate impact experiments performed in brittle materials is very compute-intensive. The cost increases even more if multiple simulations are needed to account for the randomness in crack length, location, and orientation, which is inherently found in real-world materials. Constructing a machine learning emulator can make the process faster by orders of magnitude. There has been little work, however, on assessing the error associated with their predictions. Estimating these errors is imperative for meaningful overall uncertainty quantification. In this work, we extend the heteroscedastic uncertainty estimates to bound a multiple output machine learning emulator. We find that the response prediction is accurate within its predicted errors, but with a somewhat conservative estimate of uncertainty.

1. Introduction

Emulators based on machine learning (ML) have proven to be powerful tools for prediction. These techniques can decode complex coupling between variables, producing surrogates that require substantially reduced computational resources. Nevertheless, ML emulators, particularly those built using deep learning, have been strongly criticized because of their "black box" character, resulting in their cautious adoption in computational science. This criticism is ameliorated by uncertainty quantification (UQ) methods that can bound the calculation errors. Moreover, in numerous fields, such as medicine or engineering, it is essential to be able to estimate the level of confidence in the predictions of the model [1]. Uncertainty quantification encompasses a series of mathematical techniques that characterize the space of outcomes of a model while considering that not all the parameters are precisely specified. As such, sensitivity analysis, data assimilation, design of experiments, and risk assessment are important applications of UQ [2]. Different techniques have been developed to try to overcome the difficulty of assessing uncertainties for complex ML emulators, including dropout [3] and deep ensembles [4], and for a diverse range of applications such as PDEs [5], stochastic PDEs [6], multiscale methods [7] and computer vision [8]. In this work, we extend tools to characterize heteroscedastic uncertainty [8] to the analysis of ML emulators with multiple outputs.

ML emulators have been successfully used for applications such as classification [9], regression [10], bridging scales [11–13], and dimensionality reduction [14] problems. In recent years, there has been substantial growth in ML application for material science [15], in particular for bridging scales in fracture mechanics. One of the major challenges in this context is the discrepancy in scales between microscale cracks and the macroscale associated with bulk materials [16]. Recent work has utilized ML algorithms to study brittle material under low-strain-rate tensile dynamic loading [17–19], including recursive ML prediction for fracture behavior [20]. The evolution, growth, and interaction of cracks is key to modeling damage behavior in several brittle materials such as granite, concrete, metals, and ceramics [21–24]. A recent novel alternative is to bridge continuum and mesoscales by developing and implementing a continuum-scale effective-moduli constitutive model that is informed by crack statistics generated from the mesoscale simulations in low-strain-rate [25] or high-strain-rate [26] conditions. However, the cost of generating large

* Corresponding author.
E-mail addresses: cgarciac@lanl.gov (C. Garcia-Cardona), fernandez48@llnl.gov (M.G. Fernández-Godino), omalled@lanl.gov (D. O'Malley), tanmoy@lanl.gov (T. Bhattacharya).

https://doi.org/10.1016/j.commatsci.2021.110883
Received 22 December 2020; Received in revised form 20 August 2021; Accepted 14 September 2021
Available online 30 September 2021

Table 1
Details for the Beryllium experiential samples.

Parameter	Value
Beryllium grade	S200F
Flyer disk height	2 mm
Target disk height	4 mm
Flyer/target disk diameter	28.8 mm
Flyer/target flat within	2 μm
Flyer/target parallel within	3 μ m
Density	1.85 g/cm^3
Beryllium content	0.72 wt.%
Average grain size	11.4 μm
Polish	1 μ m diamond paste
Longitudinal wave sound speed	[13.19 – 13.20] mm/μs
Shear wave sound speed	[9.04 – 9.07] mm/μs

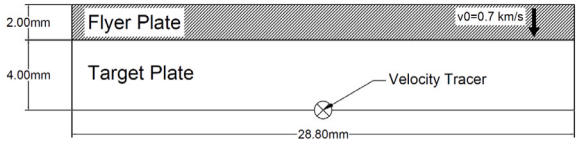


Fig. 1. Initial setup for the flyer plate test simulations. The flyer plate has an initial velocity of 0.721 km/s and it is initially in contact with the target plate.

datasets can also be prohibitive if this comes from computationally intensive high-fidelity simulations. Previous work [27] showed that an inexpensive ML emulator can inform effective moduli when trained using damage and stress information from a mesoscale model. The ML emulator can then be combined with a continuum-scale hydrodynamic simulator to make accurate predictions inexpensively. In this work, we propose a technique to assign the ML emulator uncertainty bounds that can quantify its confidence taking into account the training data variability. Since the uncertainty is rather small during most of the evolution, but is large at some specific regions, an overall measure of the expected prediction error is not the goal; rather, we model the emulator as a heteroscedastic process where the error varies depending on the input state.

2. Problem of interest: Flyer plate problem

We study the damage behavior of beryllium under dynamic loading conditions. The experiment is based on a flyer disk impact against a target disk specimen (Cady et al. [28]). The beryllium samples were machined from a vacuum hot-pressed billet of beryllium S200F grade. Table 1 shows the main details of the beryllium disk composition, machining, and dimensions.

For validation purposes, the geometry, the size, the material properties, and the initial loading of the experiments were reproduced in simulations as closely as possible. The simulations were two-dimensional and modeled a beryllium flyer plate impact against a beryllium target specimen. Fig. 1 is a schematic of the simulation setup. The flyer and target have a width of 28.8 mm, the height of the target is 4 mm, and the height of the flyer plate is 2 mm. The flyer plate has an initial vertical velocity of 0.721 km/s towards the target plate, and the total simulation time is $1.2\mu s$. A velocity tracer was placed at the middle rear of the target plate to measure the shock wave profiles, enabling comparison with the experiments and validation.

After the impact, the target plate is subject to a strong compression that later becomes tension as the shock wave travels within the material and bounces against the borders of the plate. This indirect uniaxial tensile load leads to a Mode I crack growth dominated problem (the loading is applied perpendicular to the crack). The nature of fracture in flyer plate experiments leads to a non-homogeneous damage distribution where a concentrated region of damage forms across the target plate's midspan. In contrast, the majority of the plate remains relatively undamaged. For more information on flyer plate experiments, the reader can refer to the Refs. [28,29].

HOSS model

In this work, data generated with the Hybrid Optimization Software Suite (HOSS) [30–32] was used to build the ML emulator described in Section 3. Modeling samples in HOSS is done by using discrete elements that are further divided into finite elements. The finite-discrete-element method (FDEM) included in HOSS can model the evolution of the microcrack network in high-strain rate problems. The governing equations are conservation of mass, momentum, and energy along with Newton's laws [33–38] and time-integration is done using a central difference scheme [39]. The cracks are located in the boundary of the finite elements, and often hundreds of elements are needed to model a single fracture [35]. The fine grids required, along with the explicit integration scheme, result in very expensive simulations.

HOSS high-fidelity model has been validated against experiments in a number of settings including Split Hopkinson Bar tests on granite [37], failure processes in shale [40], fracture coalescence processes in granite [41] and earthquake damage [42]. HOSS can also account for deformation in metals through a recent plasticity model [43]. HOSS explicitly accounts for crack nucleation, evolution, and coalescence. However, it does not account for microstructure, deformation twinning, dislocations, or atomic breaking at crack tips. The problem of interest in this work can be considered a pure tension problem dominated by opening failure. Still, since not every discrete element edge is oriented orthogonally to the applied load in HOSS, both shear and tearing modes occur at a local mesh element scale. The connections between finite elements are made using springs, so if two elements are pulled apart, a small space appears between them. This also allows one element to slide relative to another.

To recreate the flyer plate simulations in HOSS the inputs needed are the flyer plate mesh, the target plate mesh, the material properties (Beryllium in this case), the distribution and length of the initial cracks in the target plate and the initial velocity of the flyer plate (the target plate is stationary). HOSS is a deterministic model; hence, to obtain the statistical variability naturally existent in materials, we randomly generate the initial crack location, orientation, and length. There are 200 initial cracks, and they are only imposed in the target specimen. A uniform distribution is used to determine the initial crack location (x, y) within the target plate. The horizontal coordinate distribution corresponds to $x \sim U[0, 28.8\ \text{mm}]$ while the vertical coordinate distribution is $y \sim U[0, 4\ \text{mm}]$ (see Fig. 1). The initial orientation of the cracks (θ) follows the uniform distribution $\theta \sim U[0°, 180°)$. The initial crack lengths are determined based on a power-law probability density function [44,45], and the lengths vary between 0.1 mm and 0.3 mm. The location, length, and orientation distributions generate only the initial conditions for the crack network within the target plate and are changed randomly in every simulation. The probability density function (power-law function) used to generate the initial crack length distribution on the target plate, $f_1(a, t = 0)$, is

$$f_1(a, t = 0) = \frac{q a^{(q-1)}}{a_2^q - a_1^q}, \tag{1}$$

where $q = -3$, $a_1 = 0.1$, $a_2 = 0.3$, and a is a real number in the range $[a_1, a_2]$. The end values a_1 and a_2 are the initial minimum and maximum crack length. Therefore,

$$f_1(a, t = 0) \approx \frac{a^{-4}}{321}. \tag{2}$$

The evolution of the crack network takes place within the HOSS simulations. The time-dependent crack probability density function is obtained from each HOSS simulation after its completion. Each simulation spans 1.2 μs using HOSS time steps of 10^{-5} μs, with outputs every 0.0025 μs, which we refer to as a time step—each simulation is thus 480 time-steps long. Fig. 2 shows the shock wave velocity at the initial, intermediate and final time ($t = 0$, $t = 0.6$ μs, $t = 1.2$ μs, respectively) for a HOSS simulation of the flyer plate problem. At $t = 1.2$ μs, it is

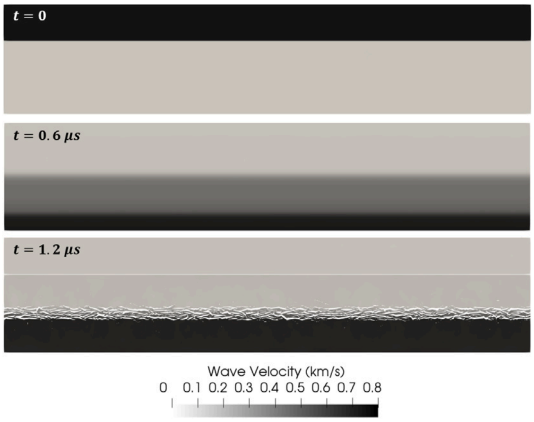


Fig. 2. HOSS simulation. Shock wave velocity at initial, intermediate and final times.

observed that the failure is produced in the middle section of the target plate because the target plate is twice the height of the flyer plate (see Fig. 1). Note that nucleation and coalescence of cracks occur as the simulation proceeds.

A single simulation of the flyer plate problem, requires 87,066 elements in HOSS to describe the crack dynamics accurately. Each simulation takes 160-CPU hours and produces 23 GBs of simulation data. Available data for this flyer plate problem includes a total of 100 HOSS simulations. The HOSS computational cost associated is prohibitive for many applications where multiple simulations are needed, such as optimization and UQ, leading to the need for cheaper ML emulators.

3. Uncertainty quantification of machine learning emulator

The goal of UQ is to assign a level of confidence to ML emulator predictions. In turn, this assignment can be used to estimate decision risks or provide a principled selection mechanism of the most impactful experiments: the ones that minimize the uncertainty. In the case of ML emulators, the UQ approaches are designed to predict both the regular input–output mapping between features and quantities of interest, together with an additional set of outputs intended to capture the level of confidence in the ML emulator predictions. Here we use a heteroscedastic [8] set up to represent the uncertainty in the regressor emulator. The heteroscedastic formulation characterizes the output of the regression emulator as a normally-distributed random variable with parameters that are feature-dependent, i.e., the variance of the predictions is heterogeneous and domain-dependent. In contrast with many applications that consist of only one output, we explore the case of UQ for emulators with multiple outputs. Further, we develop a framework for constructing multivariate heteroscedastic UQ bounds.

3.1. Machine learning emulator

Since simulating the evolution of the microcrack network in high-strain rate problems is computationally very expensive, ML emulators are gaining traction as alternative data-driven surrogate emulators for this application. With enough data, these emulators can synthesize the system's dynamics, achieving accuracy comparable to the high-fidelity models. At the same time, they are more computationally efficient and require much less time for evaluation.

In previous work [27], we demonstrated that a recurrent neural network (RNN) was able to predict two quantities of interest, namely the length of the longest crack, L_{long}, and the maximum tensile stress, S_{yy}, as a function of time for the flyer plate problem described in Section 2. In this work we estimate the uncertainty in the one-time-step response prediction of the coupled system formed by the same quantities of interest: L_{long} and S_{yy}. The more complex task of estimating uncertainties in a multivariate RNN emulator will be addressed in future work.

To build the emulator, the quantities of interest are paired at each time step t as

$$\left(L_{\text{long}}^{(t)}, S_{yy}^{(t)}\right). \tag{3}$$

Given a sequence of d consecutive time steps:

$$\left[\left(L_{\text{long}}^{(t-d)}, S_{yy}^{(t-d)}\right), \ldots, \left(L_{\text{long}}^{(t-2)}, S_{yy}^{(t-2)}\right), \left(L_{\text{long}}^{(t-1)}, S_{yy}^{(t-1)}\right)\right], \tag{4}$$

the emulator is trained to predict the next pair in the time sequence $\left(L_{\text{long}}^{(t)}, S_{yy}^{(t)}\right)$. Accordingly, the input patterns for the emulator conform to sequences as the one in (4), while the output consist of pairs as in (3). To simplify the notation the ith input pattern is denoted by $\mathbf{x_i}$ and the corresponding output pair is denoted by $\mathbf{y_i}$. Hence, the training dataset is denoted as the collection $\{\mathbf{x_i}, \mathbf{y_i}\}_{i=1}^{N}$, where $\mathbf{x_i} \in \mathbb{R}^{2d}$, $\mathbf{y_i} \in \mathbb{R}^2$ and N represents the number of samples in the training set.

We design the emulator as a multilayer feed-forward neural network composed of neurons with dense connections. The output o_j^i of each artificial neuron j in layer i is computed as

$$o_j^i = h\left(\mathbf{w}_j^i \cdot \boldsymbol{\xi}^i + b_j^i\right), \tag{5}$$

where $\boldsymbol{\xi}^i$ represents the input vector at layer i; $\mathbf{w}_j^i$ and b_j^i represent neuron parameters: weight vector and bias, respectively; the operator $\cdot$ denotes a dot product; and h, the activation function. The activation function used is a rectified linear unit (ReLU) and corresponds to the following operation: $\text{ReLU}(v) = \max(0, v)$.

In a feed-forward network, the information propagates layer-wise: the input vector $\boldsymbol{\xi}^i$ at layer i is constructed by concatenation of the outputs of the neurons at layer $i-1$, with layer 0 being the input layer and the last layer being the output layer, whose output constitutes the output of the network. Intermediate layers (i.e., different from the input and output layers) are called hidden layers. The overall mapping computed by the neural network can be denoted as $f(\mathbf{x})$ (or $\mathbf{f}(\mathbf{x})$ for an emulator with multiple outputs). In a supervised setup, the performance of the network is quantified by a loss function that measures the difference between the expected outputs, corresponding to the outputs of the training set $\{\mathbf{y_i}\}$, and the outputs computed by the emulator $\{\mathbf{f}(\mathbf{x_i})\}$. Training the emulator implies minimizing the loss function with respect to the emulator parameters, i.e., the weights and biases of all the neurons in the network. Section 3.2 describes the loss functions that are optimized, which are mean squared error (MSE) functions modified to consider the uncertainty prediction explicitly.[1]

3.2. Heteroscedastic approach

A heteroscedastic uncertainty estimate assigns a different uncertainty to each sample. Specifically, the heteroscedastic formulation assumes that the prediction can be modeled as a normal random variable with domain-dependent parameters. For a regression emulator of only one output, the ML emulator learns two outputs: the regular regression prediction $f(\mathbf{x_i})$, which corresponds to the mean, and the uncertainty represented by an additional output $\sigma(\mathbf{x_i})^2$, which corresponds to the variance of the normal distribution. The emulator is trained by minimizing the heteroscedastic loss over the entire dataset,

$$\mathcal{L}(y, f, \sigma) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\sigma(\mathbf{x_i})^2} \left\|y_i - f(\mathbf{x_i})\right\|^2 + \frac{1}{2}\log \sigma(\mathbf{x_i})^2. \tag{6}$$

This loss function, patterned after the negative log-likelihood of an univariate normal distribution, is composed of two terms: a first term that

[1] Note that in order to evaluate the UQ loss function, the output layer of the model has to be modified as well to include the additional outputs that encode the uncertainty estimation.

is a weighted mean squared error (MSE), where large error predictions are compensated by large variances, and a second term that penalizes those large variances. Note that the loss function uses the fact that the expected output y_i for input $\mathbf{x_i}$ is known, allowing for an error term, while the uncertainty σ_i is learned indirectly via regularization, i.e. the penalization term that prevents the estimation of variances that are non commensurate with the mean predictions. The balance between the first and second loss terms allows determining an 'optimal' uncertainty prediction. Hence, while the output prediction can exploit the available data explicitly, the uncertainty prediction exploits it implicitly. In some cases, this implicit estimation may correspond to a much harder task, and underlying assumptions, like model smoothness, play a greater role in determining the kind of functions that can be induced as uncertainty estimators under this formulation.

3.2.1. Multivariate approach

When the prediction includes multiple outputs, a one-dimensional normal random variable approach may not be enough to capture the influence of each in the uncertainty prediction of the others. We now show that a multivariate approach that captures the dependence of the two main variables on each other provides a better estimate of the uncertainty. To this end, it is necessary to adapt the formulation of the heteroscedastic loss to consider the probability density function (PDF) of a multivariate normal distribution. The multivariate normal distribution PDF can be written as

$$\mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{\det(\Sigma)^{-1/2}}{2\pi^{k/2}} \exp\left(-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu})\right), \tag{7}$$

where $\boldsymbol{\mu} \in \mathbb{R}^k$ stands for the mean, $\Sigma \in \mathbb{R}^{k\times k}$ represents the positive covariance matrix, and k is the space dimension. The multivariate heteroscedastic loss is expressed then as the negative log-likelihood (NLL) of the multivariate PDF with parameters promoted to functions of the input,

$$\mathcal{L}(\mathbf{y}, \mathbf{f}, \Sigma) = \frac{1}{N}\sum_{i=1}^{N}\left((\mathbf{y_i}-\mathbf{f}(\mathbf{x_i}))^T \Sigma(\mathbf{x_i})^{-1}(\mathbf{y_i}-\mathbf{f}(\mathbf{x_i})) + \log\det(\Sigma(\mathbf{x_i}))\right), \tag{8}$$

where constants $(2\pi)^{-k/2}$ and $1/2$, have been omitted because they do not change the optimum.

Correspondingly, this approach has to be designed to predict a vector $\mathbf{f}(\mathbf{x_i}) \in \mathbb{R}^k$, with the k outputs of the regular emulator and a matrix $\Sigma(\mathbf{x_i}) \in \mathbb{R}^{k\times k}$ representing the covariance matrix for the UQ estimation.

The critical component of the multivariate heteroscedastic approach is being able to guarantee that the covariance matrix, Σ, is symmetric and positive definite. An efficient strategy to achieve this is to formulate the learning task such that a matrix $A^T A$, which is positive by definition, is learned instead. This is a general strategy that can be applied to produce uncertainty estimates for ML emulators with two or more outputs.

3.2.2. The two-outputs case

In this section, we show the specific structure of the task for a two-output emulator, since this is the case of interest for this work.

The heteroscedastic UQ for an emulator with two outputs has the following structure:

- Base outputs: two to predict the mapping $\mathbf{f}(\mathbf{x_i}) \in \mathbb{R}^2$,
- Additional outputs: four to predict the components of matrix $A(\mathbf{x_i}) \in \mathbb{R}^{2\times 2}$. For simplicity, the explicit $\mathbf{x_i}$ dependence of A is (mostly) omitted in the following description.

The 2×2 matrix A can be represented in terms of scalar components a, b, c, d, as

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

which in turn yields the covariance matrix

$$\Sigma = A^T A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^T \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a^2+c^2 & ab+cd \\ ab+cd & b^2+d^2 \end{pmatrix} \triangleq \begin{pmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{12} & \sigma_{22}^2 \end{pmatrix}. \tag{9}$$

Since any simultaneous rotation of the vectors given by the columns of matrix A by the same angle leaves the covariance matrix unchanged, we choose rotation angle $\theta = \arctan(c-b)/(a+d)$, to force $b = c$. Thus, the UQ approach is built to have only five outputs: two for $\mathbf{f}(\mathbf{x_i})$ and three for the distinct components of matrix $\Sigma(\mathbf{x_i})$: $\sigma_{11}(\mathbf{x_i}), \sigma_{12}(\mathbf{x_i}), \sigma_{22}(\mathbf{x_i})$.

Further simplifications can be achieved by applying the following observations. For a positive definite covariance matrix $\Sigma \in \mathbb{R}^{2\times 2}$, the inverse can be computed analytically as

$$\Sigma^{-1} = \frac{1}{\det(\Sigma)} \begin{pmatrix} \sigma_{22}^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_{11}^2 \end{pmatrix}, \tag{10}$$

with $\det(\Sigma) = \sigma_{11}^2\sigma_{22}^2 - \sigma_{12}^2 \neq 0$. Defining: $\mathbf{e} = (e_1, e_2)^T = \mathbf{y} - \mathbf{f}(\mathbf{x})$, i.e., the difference between the ground truth $\mathbf{y}$ and the mean prediction $\mathbf{f}(\mathbf{x})$, allows to write

$$\begin{aligned}\text{NLL} &= \frac{1}{\det(\Sigma)} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}^T \begin{pmatrix} \sigma_{22}^2 - \sigma_{12} \\ -\sigma_{12}\sigma_{11}^2 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} + \log\det(\Sigma) \\ &= \frac{1}{\det(\Sigma)}(\sigma_{22}^2\, e_1^2 - 2\,\sigma_{12}\, e_1\, e_2 + \sigma_{11}^2\, e_2^2) + \log\det(\Sigma). \end{aligned} \tag{11}$$

Hence, the loss function for a heteroscedastic approach with two outputs corresponds to

$$\begin{aligned}\mathcal{L}(\mathbf{y}, \mathbf{f}, \Sigma) = \frac{1}{N}\sum_{i=1}^{N}\Bigg(&\frac{1}{\det(\Sigma(\mathbf{x_i}))}(\sigma_{22}(\mathbf{x_i})^2(y_1 - f_1(\mathbf{x_i}))^2 \\ &- 2\,\sigma_{12}(\mathbf{x_i})(y_1 - f_1(\mathbf{x_i}))(y_2 - f_2(\mathbf{x_i})) \\ &+ \sigma_{11}(\mathbf{x_i})^2\,(y_2 - f_2(\mathbf{x_i}))^2) + \log\det(\Sigma(\mathbf{x_i}))\Bigg). \end{aligned} \tag{12}$$

4. Application to the flyer plate problem

Remember that the focus of this work is to train a ML emulator to estimate the uncertainty in the one-time-step response prediction of two quantities of interest for the flyer plate problem described in Section 2. For this purpose, a feed-forward neural network was constructed, and its performance was compared against different experimental setups.

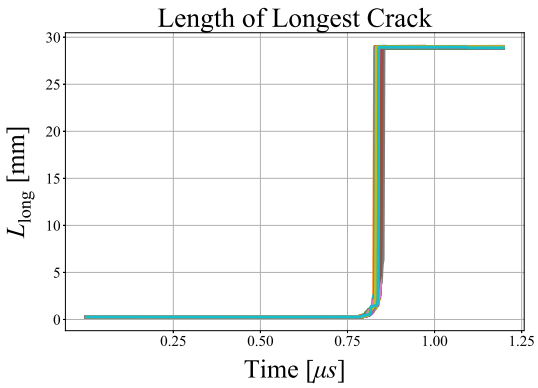
The input of the emulator constructed in this work corresponds to a sequence of d pairs of consecutive time steps of the length of the longest crack, L_{long}, and the maximum tensile stress, S_{yy}. The emulator predicts the next step in the evolution of these two quantities as well as the associated uncertainties. In other words, given the input represented by the $2d$ sequence of time steps $(1, \ldots, d)$ of the length of the longest crack and the maximum tensile stress,

$$\left[\left(L_{\text{long}}^{(1)}, S_{yy}^{(1)}\right), \left(L_{\text{long}}^{(2)}, S_{yy}^{(2)}\right), \ldots, \left(L_{\text{long}}^{(d)}, S_{yy}^{(d)}\right)\right], \tag{13}$$

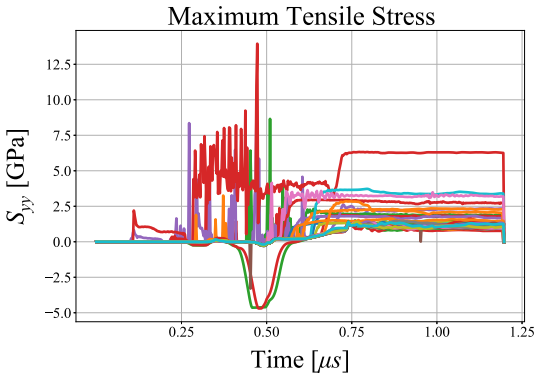
the emulator predicts: $\left(L_{\text{long}}^{(d+1)}, S_{yy}^{(d+1)}\right)$, as well as the covariance matrix $\Sigma^{(d+1)}$ for these two quantities, which is composed of variances: $\sigma^2_{L_{\text{long}}}{}^{(d+1)}$ and $\sigma^2_{S_{yy}}{}^{(d+1)}$, and covariance: $\sigma_{L_{\text{long}},S_{yy}}{}^{(d+1)}$. The window is advanced by one time step, such that the input corresponds now to quantities between $t = 2$ and $d + 1$ and the next time step $d + 2$ is predicted. The process of advancing the input window is repeated until the whole evolution sequence is predicted.

4.1. Simulation data

Similarly to [27], 100 HOSS flyer plate simulations described in were used. Each simulation includes time series of 480 time steps for different quantities of interest. From each of the 100 simulations the outputs of interest, L_{long} and S_{yy}, are extracted. Figs. 3(a) and 3(b)



(a) Evolution of length of the longest crack L_{long}.



(b) Evolution of maximum tensile stress S_{yy}.

Fig. 3. The 100 different HOSS simulations used for train and test the ML emulator.

Table 2
Datasets for one-time-step prediction.

Embedding dimension	Subsequences	Training set	Testing set
d = **10**	469	37,520	9,380
d = **20**	459	36,720	9,180
d = **30**	449	35,920	8,980

show L_{long} and S_{yy}, respectively as a function of time. As Fig. 3 shows, the length of the longest crack as a function of time is less sensitive to the considered variations in the inputs than the maximum tensile stress as a function of time.

Since the task is one-time-step prediction, each time series is split into subsequences containing $d+1$ time steps each, where the first d constitutes the input pattern $\mathbf{x}$ and the last the output $\mathbf{y}$. Remember that because we are interested in the L_{long} and S_{yy} interaction, the input subsequences are paired as in Eq. (4), while the output consists of pairs as in Eq. (3), with dimensionalities $\mathbf{x} \in \mathbb{R}^{2d}$ and $\mathbf{y} \in \mathbb{R}^2$, respectively. To guarantee that subsequences of the same simulation are not mixed during training and testing, they are first split randomly into 80% simulations for training and 20% simulations for testing. Afterward, they are further split into subsequences to build the corresponding sets, keeping subsequences of the training simulations in the training set and subsequences of the testing simulations in the testing set. Different embedding dimensions d are used. The sizes of the resulting sets are summarized in Table 2. Note that d input components are needed to predict the $d+1$ component, meaning that some of the left-most values in the simulation do not have enough previous elements to build the input pattern. Consequently, the resulting number of usable subsequences slightly decreases when d increases.

Table 3
R^2: Mean ± standard deviation for two-output models.

Embedding dimension	R^2	
	L_{long}	S_{yy}
d = **10**	0.97 ± 0.03	0.87 ± 0.22
d = **20**	0.97 ± 0.02	0.89 ± 0.09
d = **30**	0.95 ± 0.05	0.86 ± 0.10

Although a set of 100 HOSS flyer plate simulations may be perceived as a relatively small dataset, we remark that its decomposition into one-time-step predictions generates a considerable amount of training data as shown in Table 2. Previous results obtained by our group evidence that heteroscedastic UQ models have good performance even in the low data limit [46] but a full analysis of the data requirements is beyond the scope of this work.

4.2. Model architecture and training

A feed-forward neural network with two hidden layers of 200 neurons each and output layer of five neurons was constructed. A different emulator was trained for each of the different embedding dimensions. The corresponding networks have 45,405; 49,405; and 53,405 parameters, respectively. Each emulator is trained for 50 epochs with batch size of 20 using an Adam optimizer. The neural network emulators were built and trained with the Python package Keras [47]. Training one of the machine learning emulator models with the 2D heteroscedastic approach takes about 425s (7.1 min) and evaluating the testing set about 0.5s in a MacBook Pro (2.4 GHz 8-Core Intel Core i9) using CPU only. The training process is repeated 20 times, using different training–testing partitions.

Note that we did not attempt to optimize the architecture of the emulator, we focused instead on assessing the efficacy of the multivariate heteroscedastic formulation for models that have reasonable performance in the basic prediction task (which is evidenced by the relatively high R^2 values obtained as described in following sections) and that exhibit stable convergence for multiple random initializations.

4.3. Model performance

To quantify the emulator performance, the coefficient of determination (R^2),

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}\,, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i\,, \tag{14}$$

was used, as computed by the Scikit-learn Python package [48] and reported in Table 3 as the average and standard deviation over the 20 repetitions evaluated in the testing set (i.e. the set held out during training). As the table shows, the performance in terms of R^2 is good, specially for L_{long}, and the predictions with embedding dimension d = 20 is slightly better than the other two cases.

These performance results and the low training/testing times required, together with the performance achieved by other neural network-based models such as [27], demonstrate that the ML emulator approach provides good accuracy with a significant speed-up gain after the training process is complete. Ultimately the goal of this work is to accelerate uncertainty quantification workflows where the number of model runs greatly exceeds the number of model runs used in training. In such cases, this approach will accelerate the workflow even when the cost of the training data is included.

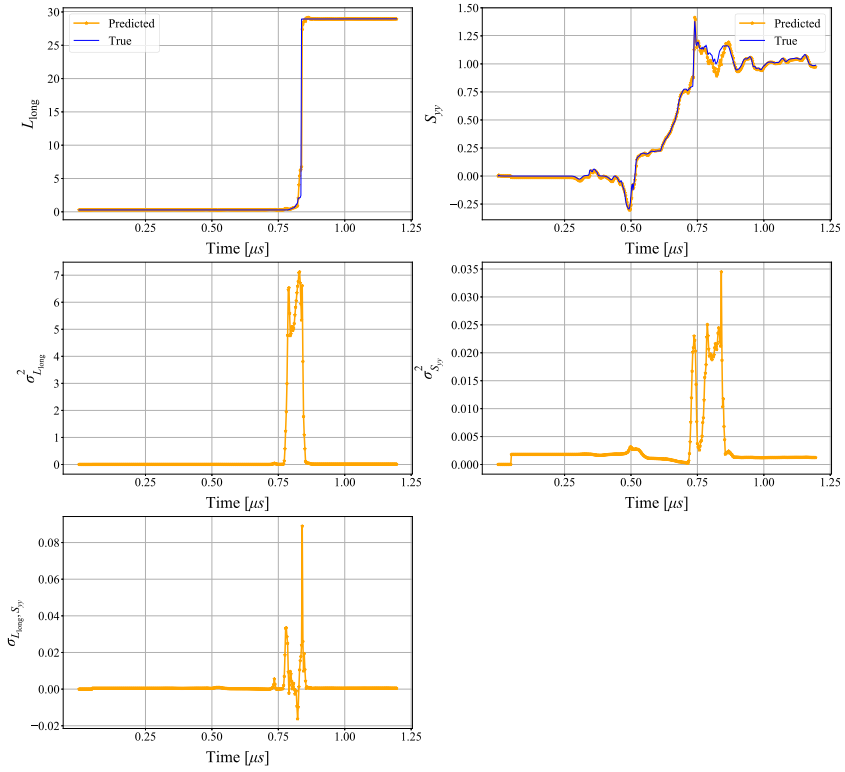


Fig. 4. Mean and covariances predicted for one of the testing series for one of the trained models for $d = 20$.

Estimating the emulator coverage

The R^2 score only takes into account the predicted values $\mathbf{f}(\mathbf{x})$, so another metric is required to evaluate the predicted uncertainty. For this purpose, we estimate the emulator coverage, which evaluates how many times the prediction falls inside the confidence interval corresponding to a specified ventile.[2] The closer the fraction of points inside the interval and the ventile are, the tighter the predicted uncertainties.

To estimate the emulator coverage, it is necessary to calculate the expected fraction inside specified contour levels of the multivariate normal distribution learned by the emulator. Contours of the multivariate normal distribution are the set of values where the argument of the exponential in the PDF is the same. As described in Appendix, each contour corresponds to an ellipse for data in $\mathbb{R}^2$, which can be expressed as

$$\gamma = (\mathbf{z}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{z}-\boldsymbol{\mu}) \,, \quad \gamma = -2\ln(1-\alpha) \,,$$

with γ corresponding to the level of the contour and α to the confidence level.

4.4. Results

To understand the need of the multivariate approach, we first demonstrate the one-variable heteroscedastic approach that independently estimates the standard deviation for the quantities of interest: length of the longest crack, L_{long}, and maximum tensile stress, S_{yy}, in the current setting. Note that the architecture of this one-variable model is similar to the two-output case, however the output layer includes four outputs for the two means and the two variances of the quantities of interest and the loss function used for training is a sum of Eq. (6) applied to each of them individually. Table 4 reports the corresponding R^2 for 20 different model realizations. This table shows similar levels of R^2 for both independent predictions, but these one-output models cannot be used to quantify the interactions between the variables of interest.

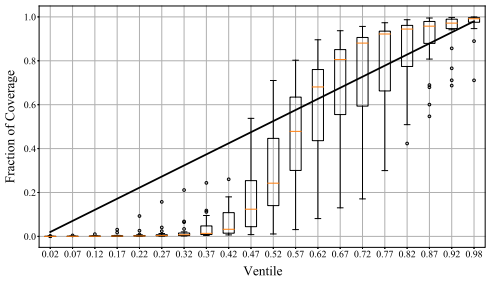
Table 4
R^2: Mean ± standard deviation for one-output models.

Embedding dimension	R^2	
	L_{long}	S_{yy}
d = **10**	0.96 ± 0.04	0.98 ± 0.004
d = **20**	0.93 ± 0.11	0.96 ± 0.049
d = **30**	0.93 ± 0.06	0.98 ± 0.004

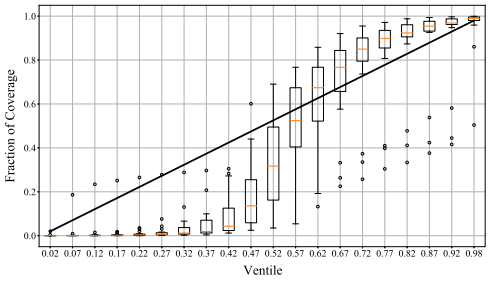
The standard deviations can be used to compute the expected distribution ventiles, which in turn can be compared to the fraction of ground truth samples that effectively fall in the given ventile. Since, in this case, the analysis is carried out independently, both independent conditions are checked simultaneously, *i.e.,* a sample is said to fall in a given ventile V if and only if L_{long_j} falls in ventile $V_{L_{\text{long}}}$ computed for the distribution of L_{long} and S_{yy_j} falls in ventile $V_{S_{yy}}$ computed for the distribution of S_{yy}. Fig. 5 shows the fraction of ground truth samples effectively falling in a given ventile. This is represented by box plots, based on 20 repetitions, for the emulator coverage computed on the testing sets as a function of the specified ventile. The meaning of each box plot is as follows: the box is plotted between the first and third quartiles, the orange line inside the box is the median, the difference between the third and the first quartile is the interquartile range (IQR). The whiskers extend between a distance of 1.5 times the IQR below the lower quartile and a distance of 1.5 times the IQR above the upper quartile. Other observed points outside the whiskers are plotted as outliers. Additionally, the ideal relationship between fraction of coverage and ventile is plotted as a continuous black line. It can be seen that in all cases the fraction of coverage exhibits an S-shape with slight over-prediction in larger ventiles and significant under-prediction in smaller ventiles.

The underprediction in the lower ventiles is a significant concern, and could result from a correlation between the two quantities. And,

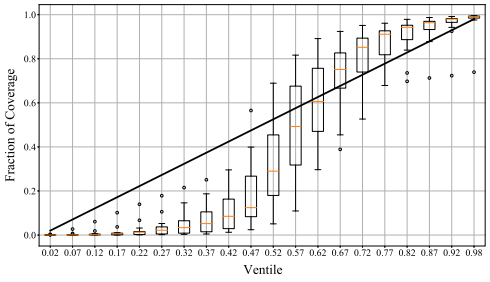
[2] The x^{th} ventile bounds the region where x-twentieth of the data are predicted to lie [49].



(a) $d = 10$.

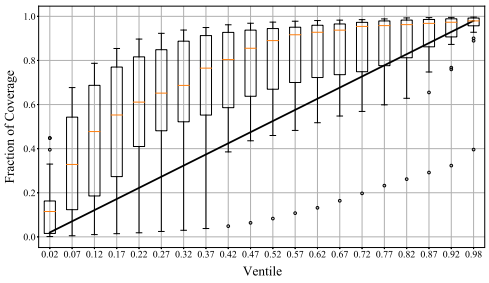


(b) $d = 20$.

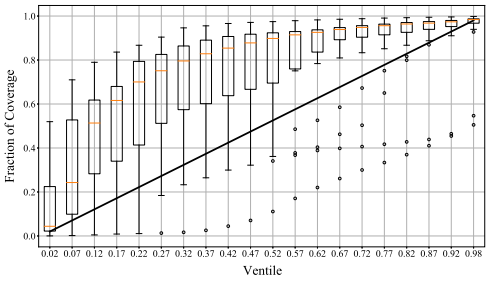


(c) $d = 30$.

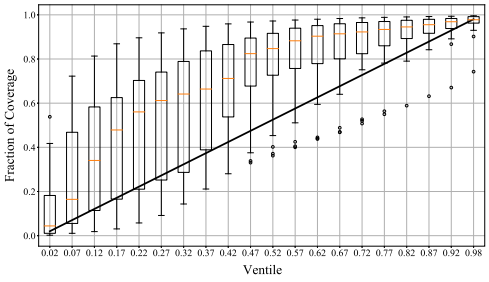
Fig. 5. Box plots of fraction of coverage for $d = 10, 20, 30$ over 20 repetitions of the one-variable heteroscedastic approach.



(a) $d = 10$.



(b) $d = 20$.



(c) $d = 30$.

Fig. 6. Box plots of fraction of coverage for $d = 10, 20, 30$ over 20 repetitions of the multivariable heteroscedastic approach.

indeed, a quantitative analysis of the covariance matrix resulting from the multivariate analysis and plotted in Fig. 4 shows that the larger scale is observed for the variance of the length of the longest crack $\sigma^2_{L_{\text{long}}}$, but this is manifest only in a window surrounding the sharp transition in the crack length. On the other hand, the scales of the variance for the maximum tensile stress $\sigma^2_{S_{yy}}$ and the covariance of length of the longest crack and maximum tensile stress $\sigma_{L_{\text{long}},S_{yy}}$ are comparable, although smaller than $\sigma^2_{L_{\text{long}}}$. The covariance $\sigma_{L_{\text{long}},S_{yy}}$ is also apparent in a window surrounding the sharp transition in the crack length. Qualitatively, this behavior seems consistent, in that the uncertainty in the length of the longest crack is concentrated in the transition region, and that the significant interaction between the variables is also concentrated in the same region.

As a result, the uncertainty estimates are qualitatively different from the one produced by the multivariate approach, where the variable correlations are captured. Fig. 6 displays the evaluation of the emulator coverage in this latter case. It can be seen that all the emulators tend to overestimate the confidence interval, which implies that the uncertainty estimation is conservative, i.e., biased towards the safer side of including more fraction of predictions than what could be inferred from the ventile. Also, the medians observed tend to be closer to the ideal for $d = 10$ and $d = 30$ than for $d = 20$, with the best overall statistics for $d = 30$. This illustrates the tension between prediction and uncertainty estimation: the ranking of emulators by R^2 may differ from the one obtained by coverage. In practice, we observe that the complexity of the function required for accurate mean prediction may be different to the complexity required for accurate variance prediction. Hence, it may be beneficial to further tune the regularization of the variance in order to avoid an oversmoothed mean prediction or limit the noise in the variance prediction.

5. Conclusion

Quantifying the uncertainty sources associated with physical models is of high importance for their credibility. When a machine learning emulator is used to speed up the process of predicting crack evolution in high-strain brittle experiments, it is important to evaluate the uncertainty associated with how well the machine learning emulator captures the underlying simulation. To some extent, this mitigates the undesirable "black box" nature of machine learning emulators. It does not make the predictions interpretable, but instead gives an indication of the expected accuracy of the prediction.

The main contribution of this work is to use machine-learning itself to bound the multivariate response of such an emulator using a heteroscedastic approach. The machine learning response is accurate within its predicted errors, while uncertainty predictions conservatively overestimate the coverage for the given confidence levels. Thus, for example, the 95% confidence interval covers about 97.6% of the data. This behavior is much more desirable than underpredicting the uncertainty. Underpredicting the uncertainty would make the predictions

seem more accurate than they are, which could have serious undesirable consequences in contexts where safety relies on the material response. The underlying cause of this overprediction is probably related to insufficiency of model assumptions – especially near a failure point – and points to the need for a nonparametric estimator for the uncertainty. We leave that development to future work.

CRediT authorship contribution statement

Cristina Garcia-Cardona: Conceptualization, Methodology, Software, Formal analysis, Visualization, Writing – original draft, Writing – Review & Editing. **M. Giselle Fernández-Godino:** Conceptualization, Investigation, Software, Formal analysis, Visualization, Writing – Original Draft, Writing – Review & Editing. **Daniel O'Malley:** Conceptualization, Writing – Review & Editing. **Tanmoy Bhattacharya:** Conceptualization, Methodology, Supervision, Project administration, Writing – Review & Editing.

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.

Data availability

Available from the authors upon request.

Acknowledgments

The authors are grateful to the anonymous reviewers, whose comments and suggestions helped improve the clarity of the manuscript.

MGFG and DO acknowledge support from the National Nuclear Security Administration's, United States Advanced Simulation and Computing program. This work has been supported in part by the Joint Design of Advanced Computing Solutions for Cancer (JDACS4C) program established by the U.S. Department of Energy (DOE) and the National Cancer Institute (NCI), United States of the National Institutes of Health, and was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and Los Alamos National Laboratory, United States under Contract DE-AC5206NA25396. Approved for public release LA-UR-20-30015 and LLNL-JRNL-817876.

Appendix. Determination of contours for multivariate normal distribution

Contours of the multivariate normal distribution are the set of values where the argument of the exponential in the PDF is the same. Therefore, contours correspond to

$$(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \gamma\,,$$

where $\gamma > 0$ is a constant value. For data in $\mathbb{R}^2$, each contour corresponds to an ellipse. For simplicity it is assumed that $\boldsymbol{\mu} = \mathbf{0}$ and that the covariance matrix has been diagonalized. Therefore,

$$(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2 .$$

Integrating the PDF of the multivariate inside the ellipse and requesting it to be equal to a specific coverage α yields to

$$4\int_0^{\sigma_x}\int_0^{\sigma_y\sqrt{\gamma-\left(\frac{x}{\sigma_x}\right)^2}} \frac{1}{2\pi}\,\frac{1}{\sigma_x\sigma_y}\times \exp\left[-\frac{1}{2}\left(\left(\frac{x}{\sigma_x}\right)^2+\left(\frac{y}{\sigma_y}\right)^2\right)\right] dy\,dx = \alpha\,,$$

where the integral is computed over the quarter ellipse in the first quadrant.

Making the change of variables to

$$\begin{aligned} x &= \sigma_x\, z\, \cos\theta \\ y &= \sigma_y\, z\, \sin\theta \end{aligned},$$

and computing the Jacobian of the transformation

$$J(z,\theta) = \begin{pmatrix} \frac{\partial x}{\partial z} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial z} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{pmatrix} \sigma_x\,\cos\theta & -\sigma_x\, z\,\sin\theta \\ \sigma_y\,\sin\theta & \sigma_y\, z\,\cos\theta \end{pmatrix},$$

and its determinant

$$\det J(z,\theta) = \sigma_x\sigma_y z\cos^2\theta + \sigma_x\sigma_y z\sin^2\theta = \sigma_x\sigma_y z\,,$$

allows for the following substitutions

$$\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2 = \left(\frac{\sigma_x\, z\,\cos\theta}{\sigma_x}\right)^2 + \left(\frac{\sigma_y\, z\,\sin\theta}{\sigma_y}\right)^2 = z^2\,,$$

$$dx\,dy = \det J(z,\theta)\,dz\,d\theta = \sigma_x\sigma_y z\,dz\,d\theta\,.$$

This, in turn, leads to

$$\alpha = \frac{1}{2\pi}\int_0^{\sqrt{\gamma}} z\,dz\,e^{\frac{-z^2}{2}}\int_0^{2\pi} d\theta = \int_0^{\sqrt{\gamma}} z e^{\frac{-z^2}{2}}\,dz\,.$$

Substituting: $s = -z^2/2$, correspondingly $ds = -z\,dz$, yields

$$\int_0^{\sqrt{\gamma}} z e^{\frac{-z^2}{2}}\,dz = \int_{-\frac{\gamma}{2}}^{0} e^s ds = \left(1 - e^{\frac{-\gamma}{2}}\right).$$

Then,

$$\begin{aligned} \alpha &= 1 - e^{\frac{-\gamma}{2}} \\ \gamma &= -2\ln(1-\alpha)\,. \end{aligned}$$

References

[1] E. Begoli, T. Bhattacharya, D. Kusnezov, The need for uncertainty quantification in machine-assisted medical decision making, Nat. Mach. Intell. 1 (2019) 20–23, http://dx.doi.org/10.1038/s42256-018-0004-1.

[2] R.C. Smith, Uncertainty Quantification: Theory, Implementation, and Applications, Vol. 12, SIAM, 2013.

[3] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: M.F. Balcan, K.Q. Weinberger (Eds.), Proceedings of Machine Learning Research, Vol. 48, PMLR, New York, New York, USA, 2016, pp. 1050–1059, URL http://proceedings.mlr.press/v48/gal16.html.

[4] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, pp. 6402–6413, URL https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[5] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, J. Comput. Phys. 397 (2019) 108850, http://dx.doi.org/10.1016/j.jcp.2019.07.048.

[6] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447, http://dx.doi.org/10.1016/j.jcp.2018.04.018.

[7] S. Chan, A.H. Elsheikh, A machine learning approach for efficient uncertainty quantification using multiscale methods, J. Comput. Phys. 354 (2018) 493–511, http://dx.doi.org/10.1016/j.jcp.2017.10.034.

[8] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, 30, Curran Associates, Inc., 2017, pp. 5574–5584, URL https://proceedings.neurips.cc/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf.

[9] S. Suthaharan, Big data classification: Problems and challenges in network intrusion prediction with machine learning, ACM SIGMETRICS Perform. Eval. Rev. 41 (4) (2014) 70–73, http://dx.doi.org/10.1145/2627534.2627557.

[10] J.-C. Huang, K.-M. Ko, M.-H. Shu, B.-M. Hsu, Application and comparison of several machine learning algorithms and their integration models in regression problems, Neural Comput. Appl. 32 (2020) 1–9, http://dx.doi.org/10.1007/s00521-019-04644-5.

[11] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017, arXiv: https://arXiv.org/abs/1711.10561 1711.10561v1 [cs.AI].

[12] Y. Cheng, J. Geng, Y. Wang, J. Li, D. Li, J. Wu, Bridging machine learning and computer network research: a survey, CCF Trans. Netw. 1 (1–4) (2019) 1–15, http://dx.doi.org/10.1007/s42045-018-0009-7.

[13] Y. Wang, S.W. Cheung, E.T. Chung, Y. Efendiev, M. Wang, Deep multiscale model learning, J. Comput. Phys. 406 (2020) 109071, http://dx.doi.org/10.1016/j.jcp.2019.109071.

[14] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D.P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Found. Trends Mach. Learn. 9 (4–5) (2016) 249–429, http://dx.doi.org/10.1561/2200000059.

[15] J. Schmidt, M.R.G. Marques, S. Botti, M.A.L. Marques, Recent advances and applications of machine learning in solid-state materials science, Npj Comput. Mater. 5 (1) (2019) 1–36, http://dx.doi.org/10.1038/s41524-019-0221-0.

[16] G. Srinivasan, J.D. Hyman, D.A. Osthus, B.A. Moore, D. O'Malley, S. Karra, E. Rougier, A.A. Hagberg, A. Hunter, H.S. Viswanathan, Quantifying topological uncertainty in fractured systems using graph theory and machine learning, Sci. Rep. 8 (1) (2018) 11665.

[17] B.A. Moore, E. Rougier, D. O'Malley, G. Srinivasan, A. Hunter, H. Viswanathan, Predictive modeling of dynamic fracture growth in brittle materials with machine learning, Comput. Mater. Sci. 148 (2018) 46–53, http://dx.doi.org/10.1016/j.commatsci.2018.01.056.

[18] A. Hunter, B.A. Moore, M. Mudunuru, V. Chau, R. Tchoua, C. Nyshadham, S. Karra, D. O'Malley, E. Rougier, H. Viswanathan, G. Srinivasan, Reduced-order modeling through machine learning and graph-theoretic approaches for brittle fracture applications, Comput. Mater. Sci. 157 (2019) 87–98, http://dx.doi.org/10.1016/j.commatsci.2018.10.036.

[19] N. Panda, D. Osthus, G. Srinivasan, D. O'Malley, V. Chau, D. Oyen, H. Godinez, Mesoscale informed parameter estimation through machine learning: A case-study in fracture modeling, J. Comput. Phys. 420 (2020) 109719, http://dx.doi.org/10.1016/j.jcp.2020.109719.

[20] M. Schwarzer, B. Rogan, Y. Ruan, Z. Song, D.Y. Lee, A.G. Percus, V.T. Chau, B.A. Moore, E. Rougier, H.S. Viswanathan, G. Srinivasan, Learning to fail: Predicting fracture evolution in brittle material models using recurrent graph convolutional neural networks, Comput. Mater. Sci. 162 (2019) 322–332.

[21] S. Meyer, E. Diegele, A. Brückner-Foit, A. Möslang, Crack interaction modelling, Fatigue Fract. Eng. Mater. Struct. 23 (4) (2000) 315–323, http://dx.doi.org/10.1046/j.1460-2695.2000.00283.x.

[22] B. Paliwal, K.T. Ramesh, An interacting micro-crack damage model for failure of brittle materials under compression, J. Mech. Phys. Solids 56 (3) (2008) 896–923, http://dx.doi.org/10.1016/j.jmps.2007.06.012.

[23] J.P. Escobedo, C.P. Trujillo, E.K. Cerreta, G.T. Gray, E.N. Brown, Effect of shock wave duration on dynamic failure of tungsten heavy alloy, J. Phys. Conf. Ser. 500 (11) (2014) 112012, http://dx.doi.org/10.1088/1742-6596/500/11/112012.

[24] F. Huq, J. Liu, A.L. Tonge, L. Graham-Brady, A micromechanics based model to predict micro-crack coalescence in brittle materials under dynamic compression, Eng. Fract. Mech. 217 (2019) 106515, http://dx.doi.org/10.1016/j.engfracmech.2019.106515.

[25] N. Vaughn, A. Kononov, B. Moore, E. Rougier, H. Viswanathan, A. Hunter, Statistically informed upscaling of damage evolution in brittle materials, Theor. Appl. Fract. Mech. 102 (2019) 210–221, http://dx.doi.org/10.1016/j.tafmec.2019.04.012.

[26] K. Larkin, E. Rougier, V. Chau, G. Srinivasan, A. Abdelkefi, A. Hunter, Scale bridging damage model for quasi-brittle metals informed with crack evolution statistics, J. Mech. Phys. Solids 138 (2020) 103921, http://dx.doi.org/10.1016/j.jmps.2020.103921.

[27] M.G. Fernández-Godino, N. Panda, K.C. Larkin, A. Hunter, D. OḾalley, R. Haftka, G. Srinivasan, Accelerating high-strain continuum-scale brittle fracture simulations with machine learning, Comput. Mater. Sci. 186 (2021) 109959, http://dx.doi.org/10.1016/j.commatsci.2020.109959.

[28] C.L. Cady, C.D. Adams, M.B. Prime, L.M. Hull, F.L. Addessio, C.A. Bronkhorst, E.N. Brown, C. Liu, T.A. Sisneros, D.W. Brown, et al., Characterization of s200-f beryllium using shock loading and quasi-static experiments, Technical Report LA-UR-11-06976, Los Alamos National Laboratory, 2011.

[29] C.M. Cady, C.D. Adams, L.M. Hull, G.T. Gray, M.B. Prime, F.L. Addessio, T.A. Wynn, P.A. Papin, E.N. Brown, Characterization of shocked beryllium, EPJ Web Conf. 26 (2012) 01009, http://dx.doi.org/10.1051/epjconf/20122601009.

[30] E. Rougier, E.E. Knight, A. Munjiza, LANL-CSM: HOSS-MUNROU technology overview, Technical Report LA-UR-13-23422, Los Alamos National Laboratory, 2013, pp. 05–10.

[31] E.E. Knight, E. Rougier, A. Munjiza, LANL-CSM: Consortium proposal for the advancement of HOSS, Technical Report LA-UR-13-23409, Los Alamos National Laboratory, 2013, pp. 05–09.

[32] E.E. Knight, E. Rougier, Z. Lei, Hybrid optimization software suite (HOSS)-educational version, Technical Report LA-UR-15-27013, Los Alamos National Laboratory, 2015.

[33] A. Munjiza, Discrete elements in transient dynamics of fractured media, (Ph.D. thesis), Swansea University, 1992.

[34] A. Munjiza, D.R.J. Owen, N. Bicanic, A combined finite-discrete element method in transient dynamics of fracturing solids, Eng. Comput. 12 (2) (1995) 145–174, http://dx.doi.org/10.1108/02644409510799532.

[35] A.A. Munjiza, The Combined Finite-Discrete Element Method, John Wiley & Sons, 2004.

[36] A.A. Munjiza, E.E. Knight, E. Rougier, Computational Mechanics of Discontinua, John Wiley & Sons, 2011.

[37] E. Rougier, E.E. Knight, S.T. Broome, A.J. Sussman, A. Munjiza, Validation of a three-dimensional finite-discrete element method using experimental results of the split hopkinson pressure bar test, Int. J. Rock Mech. Min. Sci. 70 (2014) 101–108, http://dx.doi.org/10.1016/j.ijrmms.2014.03.011.

[38] A. Munjiza, E.E. Knight, E. Rougier, Large Strain Finite Element Method: A Practical Course, John Wiley & Sons, 2015.

[39] E. Rougier, A. Munjiza, N.W.M. John, Numerical comparison of some explicit time integration schemes used in DEM, FEM/DEM and molecular dynamics, Internat. J. Numer. Methods Engrg. 61 (6) (2004) 856–879, http://dx.doi.org/10.1002/nme.1092.

[40] J.W. Carey, Z. Lei, E. Rougier, H. Mori, H. Viswanathan, Fracture-permeability behavior of shale, J. Unconv. Oil Gas Res. 11 (2015) 27–43, http://dx.doi.org/10.1016/j.juogr.2015.04.003.

[41] B. Euser, E. Rougier, Z. Lei, E.E. Knight, L.P. Frash, J.W. Carey, H. Viswanathan, A. Munjiza, Simulation of fracture coalescence in granite via the combined finite–discrete element method, Rock Mech. Rock Eng. 52 (9) (2019) 3213–3227, http://dx.doi.org/10.1007/s00603-019-01773-0.

[42] Y. Klinger, K. Okubo, A. Vallage, J. Champenois, A. Delorme, E. Rougier, Z. Lei, E.E. Knight, A. Munjiza, C. Satriano, et al., Earthquake damage patterns resolve complex rupture processes, Geophys. Res. Lett. 45 (19) (2018) 10–279, http://dx.doi.org/10.1029/2018GL078842.

[43] E. Rougier, A. Munjiza, Z. Lei, V.T. Chau, E.E. Knight, A. Hunter, G. Srinivasan, The combined plastic and discrete fracture deformation framework for finite-discrete element methods, Internat. J. Numer. Methods Engrg. 121 (5) (2020) 1020–1035, http://dx.doi.org/10.1002/nme.6255.

[44] E. Bonnet, O. Bour, N.E. Odling, P. Davy, I. Main, P. Cowie, B. Berkowitz, Scaling of fracture systems in geological media, Rev. Geophys. 39 (3) (2001) 347–383, http://dx.doi.org/10.1029/1999RG000074.

[45] S.R. Ignatovich, N.I. Bouraou, Power law of crack length distribution in the multiple damage process, Strength Mater. 51 (5) (2019) 735–745, http://dx.doi.org/10.1007/s11223-019-00122-4.

[46] C. Garcia-Cardona, Y.T. Lin, T. Bhattacharya, Uncertainty quantification for deep learning regression models in the low data limit, in: M. Papadrakakis, V. Papadopoulos, G. Stefanou (Eds.), Proceedings of 4th International Conference on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP 2021), European Community on Computational Methods in Applied Sciences (ECCOMAS), Athens, Greece, 2021, p. 19145, http://dx.doi.org/10.7712/120221.8045.19145, URL https://2021.uncecomp.org/proceedings/pdf/19145.pdf.

[47] F. Chollet, Keras documentation, 2015, URL https://keras.io/.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830, URL https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[49] R. Wilcox, Introduction To Robust Estimation and Hypothesis Testing, fourth ed., Academic Press, 2016.