Uncertainty Bounds for Multivariate Machine Learning Predictions on High-Strain Brittle Fracture

Cristina Garcia-Cardona\textsuperscript{a,*}, M. Giselle Fernández-Godino\textsuperscript{b}, Daniel O’Malley\textsuperscript{c} and Tanmoy Bhattacharya\textsuperscript{d}

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

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 [42]. Different techniques have been developed to try to overcome the difficulty of assessing uncertainties for complex ML emulators, including dropout [13] and deep ensembles [23], and for a diverse range of applications such as PDEs [48], stochastic PDEs [49], multiscale methods [6] and computer vision [19]. In this work, we extend tools to characterize heteroscedastic uncertainty [19] to the analysis of ML emulators with multiple outputs.

ML emulators have been successfully used for applications such as classification [44], regression [15], bridging scales [35, 7, 46], and dimensionality reduction [9] problems. In recent years, there has been substantial growth in ML application for material science [40], 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 [43]. Recent work has utilized ML algorithms to study brittle material under low–strain-rate tensile dynamic loading [26, 16, 33], including recursive ML prediction for fracture behavior [41]. The evolution, growth, and interaction of cracks is key to modeling damage behavior in several brittle materials such as granite, concrete, metals, and ceramics [25, 32, 10, 17]. 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 [45] or high–strain-rate [24] conditions. However, the cost of generating large datasets can also be prohibitive if this comes from computationally intensive high-fidelity simulations. Previous work [12] 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.
mation on flyer plate experiments, the reader can refer to the references [3, 4].

HOSS Model

In this work, data generated with the Hybrid Optimization Software Suite (HOSS) [36, 22, 21] 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 [27, 29, 30, 31, 37, 28] and time-integration is done using a central difference scheme [38]. The cracks are located in the boundary of the finite elements, and often hundreds of elements are needed to model a single fracture [30]. 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 [5], fracture coalescence processes in granite [11] and earthquake damage [20]. HOSS can also account for deformation in metals through a recent plasticity model [39]. 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 Figure 1). The initial orientation of the cracks (\( \theta \)) follows the uniform distribution \( \theta \sim U[0^\circ, 180^\circ] \). The initial crack lengths are determined based on a power-law probability density function [2, 18], and the lengths vary between 0.1 \( \text{mm} \) and 0.3 \( \text{mm} \). The location, length, and orientation distributions generate only the initial conditions for the crack network within the target plate.

---

Cristina Garcia-Cardona et al.: Preprint submitted to Elsevier
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

\[
f_1(a, t = 0) = \frac{qa^{q-1}}{a_1^q - a_a^q},
\]

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,

\[
f_1(a, t = 0) \approx \frac{a^{-4}}{321}.
\]

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\( \mu \)s using HOSS time steps of \( 10^{-5} \mu s \), with outputs every 0.0025\( \mu s \), which we refer to as a time step—each simulation is thus 480 time-steps long. Figure 2 shows the shock wave velocity at the initial, intermediate and final time (\( t = 0, t = 0.6\mu s, t = 1.2\mu s \), respectively) for a HOSS simulation of the flyer plate problem. At \( t = 1.2\mu s \), it is 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 Figure 1). Note that nucleation and coalescence of cracks occur as the simulation proceeds.

![Figure 2: HOSS simulation. Shock wave velocity at initial, intermediate and final times.](image)

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 \([19]\) 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 \([12]\), we demonstrated that a recurrent neural network (RNN) was able to predict two quantities of interest, namely the length of the longest crack, \( L_{\text{long}} \), 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: \( L_{\text{long}} \) 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

\[
\left( L_{\text{long}}^{(t)}, S_{yy}^{(t)} \right).
\]

Given a sequence of \( d \) consecutive time steps:

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

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 \( i \)-th input pattern is denoted by \( x_i \) and the corresponding
output pair is denoted by \( y_i \). Hence, the training data set is denoted as the collection \( \{ \mathbf{x}_i, y_i \}_{i=1}^N \), where \( \mathbf{x}_i \in \mathbb{R}^{2d}, y_i \in \mathbb{R}^2 \) 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

\[
o_j^i = h \left( \mathbf{w}_j^i \cdot \xi^i + b_j^i \right)
\]

where \( \xi^i \) 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 \( \xi^0 \) 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 constructed by the neural network can be denoted as \( f(\mathbf{x}) \) (or \( f \) 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 \( \{ y_i \} \), and the outputs computed by the emulator \( \{ f(\mathbf{x}_i) \} \). 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,

\[
\mathcal{L}(y, f, \sigma) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\sigma(\mathbf{x}_i)^2} ||y_i - f(\mathbf{x}_i)||^2 + \frac{1}{2} \log \sigma(\mathbf{x}_i)^2.
\]

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

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

where \( \mu \in \mathbb{R}^k \) 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,

\[
\mathcal{L}(y, f, \Sigma) = \\
\frac{1}{N} \sum_{i=1}^{N} \left( (y_i - f(\mathbf{x}_i))^T \Sigma(\mathbf{x}_i)^{-1}(y_i - f(\mathbf{x}_i)) \right) + \log \det(\Sigma(\mathbf{x}_i)),
\]

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 \( f(\mathbf{x}_i) \in \mathbb{R}^k \), 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, \( \Sigma \), is symmetric and positive definite. An efficient strategy to achieve this is to formulate the learning task such

---

\(^1\)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.
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 $f(x_i) \in \mathbb{R}^2$,

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

The $2 \times 2$ 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

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

(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 two outputs: two for $f(x_i)$ and three for the distinct components of matrix $A(x_i)$: $\sigma_{11}(x_i), \sigma_{12}(x_i), \sigma_{22}(x_i)$.

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

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

(10)

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

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

(11)

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

$$\mathcal{L}(y, f, \Sigma) = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{\det(\Sigma(x_i))} (\sigma_{22}(x_i)^2(y_1 - f_1(x_i))^2 \\
- 2 \sigma_{12}(x_i)(y_1 - f_1(x_i))(y_2 - f_2(x_i)) \\
+ \sigma_{11}(x_i)^2 (y_2 - f_2(x_i))^2 + \log \det(\Sigma(x_i)) \right).$$  

(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, $L_{\text{long}}$, 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,

$$\begin{pmatrix} (L_{\text{long}}^{(1)}, S_{yy}^{(1)}) \\ (L_{\text{long}}^{(2)}, S_{yy}^{(2)}) \\ \vdots \\ (L_{\text{long}}^{(d)}, S_{yy}^{(d)}) \end{pmatrix},$$

(13)

the emulator predicts: $(L_{\text{long}}^{(d+1)}, S_{yy}^{(d+1)})$, as well as the covariance matrix $\Sigma^{(d+1)}$ for these two quantities, which is composed of variances: $\sigma_{L_{\text{long}}}^2$ and $\sigma_{S_{yy}}^2$, 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 [12], 100 HOSS flyer plate simulations described in Sec. 2 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, $L_{\text{long}}$ and $S_{yy}$, are extracted. Figure 3(a) and Figure 3(b) show $L_{\text{long}}$ and $S_{yy}$, respectively as a function of time. As Figure 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 $x$ and the last the output $y$. Remember that because we are interested in the $L_{\text{long}}$ 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 $x \in \mathbb{R}^{2d}$ and $y \in \mathbb{R}^2$.
Subsequences

Training Set

Testing Set

Cristina Garcia-Cardona et al.: Preprint submitted to Elsevier
inside the confidence interval corresponding to a specified ventile. 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 A, each contour corresponds to an ellipse for data in \( \mathbb{R}^2 \), which can be expressed as

\[
\gamma = (z - \mu)^T \Sigma^{-1} (z - \mu), \quad \gamma = -2 \ln (1 - a),
\]

with \( \gamma \) corresponding to the level of the contour and \( a \) 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, \( L_{\text{long}} \), 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.

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 \( L_{\text{long}} \) falls in ventile \( V_{l_{\text{long}}} \) computed for the distribution of \( L_{\text{long}} \) and \( S_{yy} \) falls in ventile \( V_{s_{yy}} \) computed for the distribution of \( S_{yy} \). Figure 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, indeed, a quantitative analysis of the covariance matrix resulting from the multivariate analysis and plotted in Figure 4 shows that the larger scale is observed for the variance of the length of the longest crack \( \sigma_{L_{\text{long}}}^2 \), 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 \( \sigma_{S_{yy}}^2 \) and the covariance of length of the longest crack and maximum tensile stress \( \sigma_{L_{\text{long}},S_{yy}} \) are comparable, although smaller than \( \sigma_{L_{\text{long}}}^2 \). 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

---

**Table 3**

\( R^2 \): Mean ± standard deviation for two-output models.

| Embedding Dimension | \( L_{\text{long}} \)   | \( S_{yy} \)   |
|---------------------|------------------------|---------------|
| \( d = 10 \)        | 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   |

**Table 4**

\( R^2 \): Mean ± standard deviation for one-output models.

| Embedding Dimension | \( L_{\text{long}} \)   | \( S_{yy} \)   |
|---------------------|------------------------|---------------|
| \( d = 10 \)        | 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  |

---

2The \( x \)th ventile bounds the region where \( x \)-twentieth of the data are predicted to lie [47].
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. Figure 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.

Data Availability

Available from the authors upon request.

Acknowledgements

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 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) 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 under Contract DE-AC5206NA25396. Approved for public release LA-UR-20-30015 and LLNL-JRNL-817876.

References

[1] Begoli, E., Bhattacharya, T., Kusnezov, D., 2019. The need for uncertainty quantification in machine-assisted medical decision making. Nature Machine Intelligence 1, 20–23. doi:10.1038/s42256-018-0004-1.
[2] Bonnet, E., Bour, O., Odling, N.E., Davy, P., Main, I., Cowie, P., Berkowitz, B., 2001. Scaling of fracture systems in geological media. Reviews of geophysics 39, 347–383. doi:10.1029/1999RG000074.
[3] Cady, C.L., Adams, C.D., Prime, M.B., Hull, L.M., Adessio, F.L., Bronkhorst, C.A., Brown, E.N., Liu, C., Sisneros, T.A., Brown, D.W., et al., 2011. Characterization of $2000\text{-}F$ Berylhum using shock loading and quasi-static experiments. Technical Report LA-UR-11-00976, Los Alamos National Laboratory.
[4] Cady, C. M., Adams, C. D., Hull, L. M., Gray, G. T., Prime, M. B., Adessio, F. L., Wynn, T. A., Papin, P. A., Brown, E. N., 2012. Characterization of shocked berylhum. EPI Web of Conferences 26, 01009. doi:10.1051/epjconf/20122601009.
[5] Carey, J.W., Lei, Z., Rougier, E., Mori, H., Viswanathan, H., 2015. Fracture-permeability behavior of shale. Journal of unconventional oil and gas resources 11, 27–43. doi:10.1816/j.juogr.2015.04.003.
[6] Chan, S., Elsheikh, A.H., 2018. A machine learning approach for efficient uncertainty quantification using multiscale methods. Journal of Computational Physics 354, 493–511. doi:10.1016/j.jcp.2017.10.034.
[7] Cheng, Y., Geng, J., Wang, Y., Li, J., Li, D., Wu, J., 2019. Bridging machine learning and computer network research: a survey. CCF Transactions on Networking 1, 1–15. doi:10.4108/2045-819-8089-T.
[8] Chollet, F., 2015. Keras documentation. URL: https://keras.io/.
[9] Cichocki, A., Lee, N., Oseledets, I., Phan, A.H., Zhao, Q., Mandic, D.P., 2016. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. Foundations and Trends® in Machine Learning 9, 249–429. doi:10.1561/2200000059.
[10] Escobedo, J.P., Trujillo, C.P., Cerreta, E.K., Gray, G.T., Brown, E.N., 2014. Effect of shock wave duration on dynamic failure of tungsten heavy alloy. Journal of Physics: Conference Series 500, 112012. doi:10.1088/1742-6596/500/11/112012.
[11] Euser, B., Rougier, E., Lei, Z., Knight, E.E., Frash, L.P., Carey, J.W., Viswanathan, H., Munjiza, A., 2019. Simulation of fracture coalescence in granite via the combined finite–discrete element method. Rock Mechanics and Rock Engineering 52, 3213–3227. doi:10.1007/s00603-019-0777-8.
[12] Fernández-Godino, M.G., Panda, N., Larkin, K.C., Hunter, A., O’Malley, D., Haftka, R., Srinivasan, G., 2021. Accelerating high-strain continuum-scale brittle fracture simulations with machine learning. Computational Materials Science 186, 109959. doi:10.1016/j.commatsci.2020.109959.
[13] Gal, Y., Ghahramani, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: Balcan, M.F., Weinberger, K.Q. (Eds.), Proceedings of Machine Learning Research, PMLR, New York, New York, USA. pp. 1050–1059. URL: http://proceedings.mlr.press/v48/gal16.html.
[14] Garcia-Cardona, C., Lin, Y.T., Bhattacharya, T., 2021. Uncertainty quantification for deep learning regression models in the low data limit, in: Papadrakakis, M., Papadopoulos, V., Stefanou, G. (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. p. 19145. URL: https://2021.uncecomp.org/proceedings/pdf/19145.pdf.
[15] Huang, J.C., Ko, K.M., Shu, M.H., Hsu, B.M., 2020. Application and comparison of several machine learning algorithms and their integration models in regression problems. Neural Computing and Applications 32, 1–9. doi:10.1007/s00521-019-04644-5.
[16] Hunter, A., Moore, B.A., Mudunuru, M., Chau, V., Tchoua, R., Nyshadham, C., Karra, S., O’Malley, D., Rougier, E., Viswanathan, H., Srinivasan, G., 2019. Reduced-order modeling through machine learning and graph-theoretic approaches for brittle fracture applications. Computational Materials Science 157, 87–98. doi:10.1016/j.commatsci.2018.10.036.
[17] Huq, F., Liu, J., Tonge, A., Graham-Brady, L., 2019. A micromechanics based model to predict micro-crack coalescence in brittle materials under dynamic compression. Engineering Fracture Mechanics 217, 106515. doi:10.1016/j.engfracmech.2019.106515.
[18] Ignatovich, S.R., Bourau, N.L., 2019. Power law of crack length distribution in the multiple damage process. Strength of Materials 51, 735–745. doi:10.1007/s11203-019-00122-4.
[19] Kendall, A., Gal, Y., 2017. What uncertainties do we need in bayesian deep learning for computer vision?, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems, Curran.
A. 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

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

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 $\mu = 0$ and that the covariance matrix has been diagonalized. Therefore,

$$(x - \mu)^T \Sigma^{-1} (x - \mu) = \left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2$$
Integrating the PDF of the multivariate inside the ellipse and requesting it to be equal to a specific coverage \( \alpha \) yields to

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

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

Making the change of variables to

\[
x = \sigma_x \ z \ \cos \ \theta
\]
\[
y = \sigma_y \ z \ \sin \ \theta
\]

and computing the Jacobian of the transformation

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

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

\[
\alpha = \frac{1}{2\pi} \int_0^{\sqrt{7}} \int_0^2 z \ e^{-\frac{z^2}{2}} \ dz \ d\theta = \int_0^{\sqrt{7}} z \ e^{-\frac{z^2}{2}} \ dz.
\]

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

\[
\int_0^{\sqrt{7}} z \ e^{-\frac{z^2}{2}} \ dz = \int_{-\sqrt{2}}^0 e^s \ ds = \left( 1 - e^{-\frac{\sqrt{7}}{2}} \right).
\]

Then,

\[
\alpha = 1 - e^{-\frac{\sqrt{7}}{2}}
\]
\[
\Rightarrow \gamma = -2 \ln \left( 1 - \alpha \right).
\]