JOM, Vol. 71, No. 8, 2019
https://doi.org/10.1007/s11837-019-03572-y

MULTISCALE COMPUTATIONAL STRATEGIES FOR HETEROGENEOUS MATERIALS WITH DEFECTS: COUPLING MODELING WITH EXPERIMENTS AND UNCERTAINTY QUANTIFICATION

Predicting Microstructure-Sensitive Fatigue-Crack Path in 3D Using a Machine Learning Framework

KYLE PIERSON,[1] AOWABIN RAHMAN,[2] and ASHLEY D. SPEAR[2,3]

1.—School of Computing, University of Utah, Salt Lake City 84112, USA. 2.—Department of Mechanical Engineering, University of Utah, Salt Lake City 84112, USA 3.—e-mail: ashley.spear@utah.edu

The overarching aim of this paper is to explore the use of machine learning (ML) to predict the microstructure-sensitive evolution of a three-dimensional (3D) crack surface in a polycrystalline alloy. A convolutional neural network (CNN)-based methodology is developed to establish spatial relationships between micromechanical/microstructural features in a cyclically loaded, uncracked microstructure and the 3D crack path, the latter quantified by the vertical deviation (i.e., z-offset) of the crack along a specified axis. The proposed methodology consists of (i) a feature selection and reduction scheme to identify a lower-dimensional representation of the experimentally measured microstructure and computed micromechanical fields, which allows for computational feasibility in predicting the z-offsets; (ii) a CNN model to compute the z-offset as a function of the local, lower-dimensional feature data; and (iii) a radial basis function smoothing spline to ensure spatial continuity between the independently predicted z-offsets. The proposed CNN-based methodology is shown to improve on the accuracies obtained using existing ML models such as XGBoost and to provide a definitive way of quantifying model uncertainty associated with CNN predictions. To further investigate the applicability of ML models, multiple prediction strategies with which to deploy ML algorithms are proposed and the relative performance of ML algorithms corresponding to each prediction strategy are analyzed. The presented work thus provides a framework to find an encoded representation of 3D microstructure and micromechanical data and develop methods to predict microstructure-sensitive crack evolution based on this encoded representation, while quantifying associated prediction uncertainties.

INTRODUCTION

Fatigue cracking contributes to a majority of in-service failures of engineering structures, which are often driven by onset and accumulation of microstructurally small cracks.[1] The initiation and early-stage growth of these small cracks are dependent on microstructural features that influence, for example, crack interactions at grain boundaries.[2] Computational modeling of crack initiation and growth can be complex, partly because understanding the effects of a broad range of microstructural features on the constitutive response can be extremely difficult to achieve using conventional continuum approaches.[2] While the growth rate of long cracks can be accurately predicted using the Paris–Erdogan law (or similar relationships),[3–5] microstructurally small cracks can exhibit greater variability in their growth behavior compared with long cracks.[4] This is evidenced by the larger scatter in the growth rate of small cracks with the same nominal value of stress intensity factor.[4] Yet understanding the behavior of these small cracks is critical, as they can account for upwards of 80% to 90% of a component's fatigue lifetime.[6] As such, accurate predictions of crack behavior can improve the reliability of structural components, as well as lower associated production and maintenance costs, as described in Ref. 7.

In an attempt to model the behavior of microstructurally small cracks, both empirical and numerical approaches have been employed based on

 (Published online July 1, 2019)

certain underlying principles and influencing factors. Fatemi and Yang[8] and Hussain[5] review early phenomenological theories on small-crack behavior, in which plasticity effects, metallurgical effects, and crack closure are suggested as possible explanations for small-crack behavior. Factors such as grain shape and crystal orientation, near-neighbor distances, grain fracture toughness, and intrinsic flaw size are also considered as factors contributing to crack nucleation and early-stage propagation.[9,10] In some cases, finite-element analysis using crystal plasticity models, as originally proposed by Asaro,[11] are used in literature to investigate the effect of the spatial variability of microstructural features and micromechanical fields on small-crack behavior (e.g., see Ref. 12). A more comprehensive review of empirical and numerical approaches investigating the behavior of small cracks can be found in other literature.[5–8,13]

While the aforementioned approaches often rely on stress intensity factors[12] and/or fatigue indicator parameters[14,15] as representative mesoscopic surrogates for the driving force behind crack initiation and propagation, there is a need for a more comprehensive, general framework that accounts for the complex spatial, nonlinear relationships between the relevant features, i.e., microstructural features and micromechanical fields, and a given representation of the crack. Data-driven methods can potentially be a viable alternative approach to address this challenge, as they can leverage large, high-dimensional datasets, obtained through experiments or simulations, to model these complex relationships.[16–18] Machine learning (ML) models have already exhibited a wide range of applicability within the materials science community, including for materials discovery,[19] optimal design of experiments,[20] and image-based materials characterization.[21,22] In the context of predicting crack behavior, data-driven methods such as principal component analysis (PCA) have been employed to determine reduced-order representations that correlate with fatigue indicator parameters (FIPs).[23,24] In similar work,[25] a random forest learning algorithm was employed to predict stress hot spots that were computed from full-field crystal plasticity simulations, where the algorithm was trained using features that encode the local crystallography, geometry, and connectivity of the microstructure. Probabilistic models such as Bayesian networks,[7,26] which are nonparametric and can account for uncertainties in predictions, have also been used to compute fatigue-related parameters such as residual life and equivalent stress intensity factors, respectively.

This paper expands previous work on data-driven methods in fatigue modeling and proposes a framework using a convolutional neural network (CNN) model to predict 3D crack paths based on microstructural and micromechanical features. While previous, relevant work using data-driven approaches has primarily focused on identifying micromechanical and microstructural variables that contribute to the direction and rate of crack propagation,[26] or on mapping global variables (such as chemical composition, grain size, heat treatment, and cyclic stress intensity factor) to the one-dimensional crack growth rate,[27] the work presented herein concerns the use of CNN to quantitatively predict the local crack path, in 3D, as a function of local microstructural and micromechanical features. CNNs are particularly well suited for problems that require finding spatial, nonlinear relationships between input and a given response variable of interest and have been successfully used in related applications such as classification of microstructures based on scanning electron microscopy (SEM) images[20] and determination of material properties based on microstructure.[28,29] Prior to training the CNN model, the input features (i.e., microstructural features and micromechanical fields) are selected based on previous correlation analysis by Pierson et al.[30] (described below). PCA analysis is performed to convert the relevant input features to unique low-dimensional descriptors, whose values are specific to a given location within a microstructure. The 3D map of these descriptor values is then introduced into the CNN model. As a postprocessing step, to retain spatial continuity, a smoothing operation is performed on the predictions of crack surface elevations obtained from the CNN model.

The next subsection provides background information on the experimental and simulation data used to train and evaluate the ML model.

Prior Work by the Authors

In previous work,[30] the authors conducted a systematic correlation analysis between computed micromechanical fields in a 3D, uncracked polycrystal and the observed path of an eventual fatigue crack. Specifically, an experimentally measured volume of an Al-Mg-Si alloy[31] was modeled using a high-fidelity, concurrent-multiscale, finite-element mesh with a crystal-plasticity constitutive model.[32] Cyclic loading was simulated at a load ratio of R = 0.5 (consistent with experiment), and computed field variables (or derivatives thereof) based on stress, strain, and slip were parameterized to a regular 3D grid. A complete list of the 22 variables computed at each time step during the finite-element simulation is presented in Table I. Additionally, the cyclic change in each of the variables was computed between the peak and minimum load for each of five simulated loading cycles. Figure 1 shows six of the parameterized variables (five cyclic damage metrics and the cyclic micromechanical Taylor factor[33]). Once the cyclic change in the computed field variables was shown to converge, the spatial gradients of the variables were calculated using a finite-difference approach. In total, 88

Table I. List of 22 variables computed during finite-element simulation[30]

D_1	Maximum value of accumulated slip among 12 octahedral slip systems
D_2	Maximum value of total accumulated slip over each slip plane
D_3	Accumulated slip summed over all slip systems
D_4	Maximum value of energy dissipated on a given slip plane during plastic deformation
D_5	Modified Fatemi–Socie parameter
$\bar{\epsilon}$	Symmetric strain tensor composed of ϵ_{xx}, ϵ_{yy}, ϵ_{zz}, ϵ_{xy}, ϵ_{xz}, and ϵ_{yz}
ϵ_1	Principal eigenvalue of the strain tensor
ϵ_{vM}	von Mises strain
$\bar{\sigma}$	Symmetric stress tensor composed of σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{xz}, and σ_{yz}
σ_1	Principal eigenvalue of the stress tensor
σ_{vM}	von Mises stress
M^{micro}	Micromechanical Taylor factor

For each variable, the cyclic change (Δ), spatial gradient (∇), and spatial gradient of the cyclic value ($\Delta(\nabla)$) were determined, resulting in 88 different variables at each point in the simulation domain.

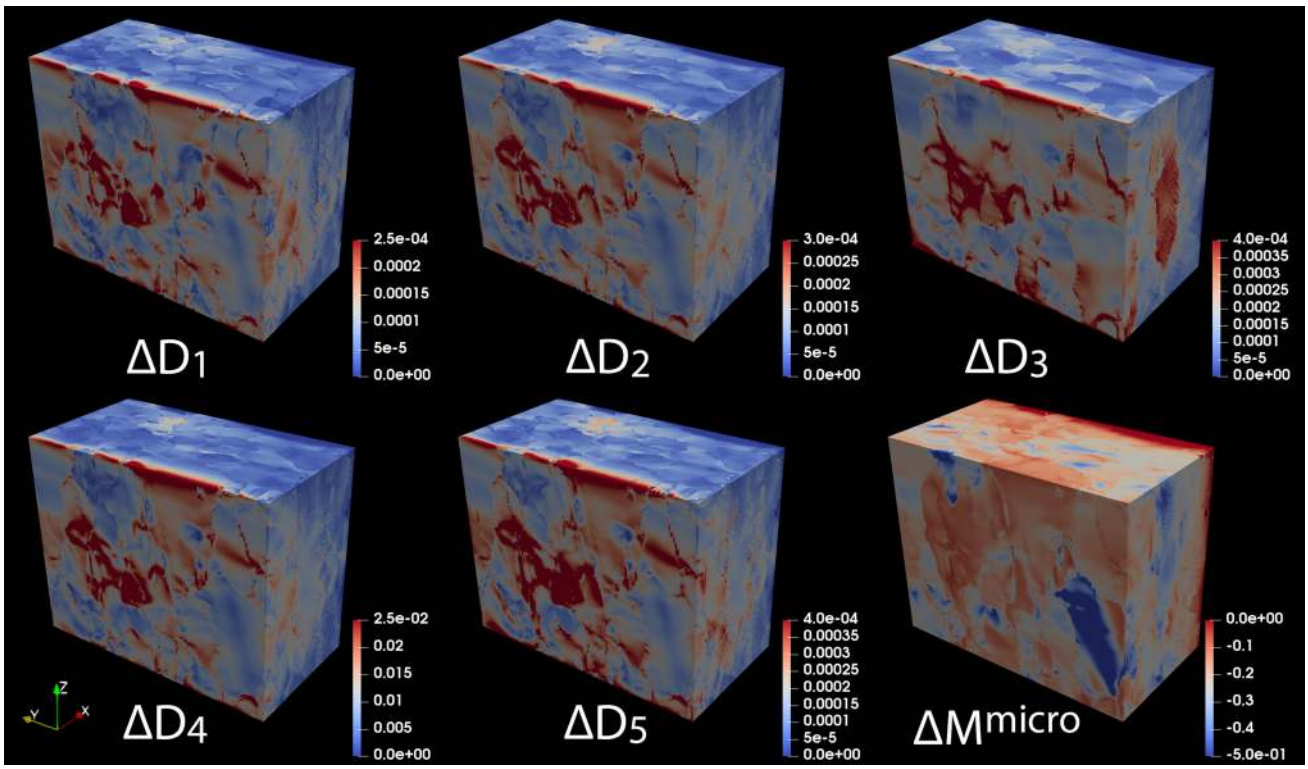

Fig. 1. Grid data showing cyclic values of slip-based damage metrics computed for uncracked microstructure in multiscale finite-element simulation. A description of each metric is provided in Table I. Reprinted with permission from Ref. 30.

variables were considered in the correlation study, of which 44 were based on spatial gradients of the micromechanical fields. The parameterized variables were then systematically correlated with distance to the known crack surface. Correlation coefficients for all 88 variables are shown in Fig. 2. In general, the spatial gradients of the micromechanical field variables (Fig. 2c and d) exhibited a stronger correlation with the crack path than did the field variables themselves (Fig. 2a and b). To assess whether the correlation coefficient values shown in Fig. 2c and d were meaningful, correlation analyses were also performed between the 88 variables and alternative paths throughout the microstructure. The correlation coefficients for the alternative paths were consistently weaker than those for the actual crack surface, suggesting that micromechanical fields of the cyclically loaded, uncracked microstructure might provide some degree of predictiveness for the microstructurally

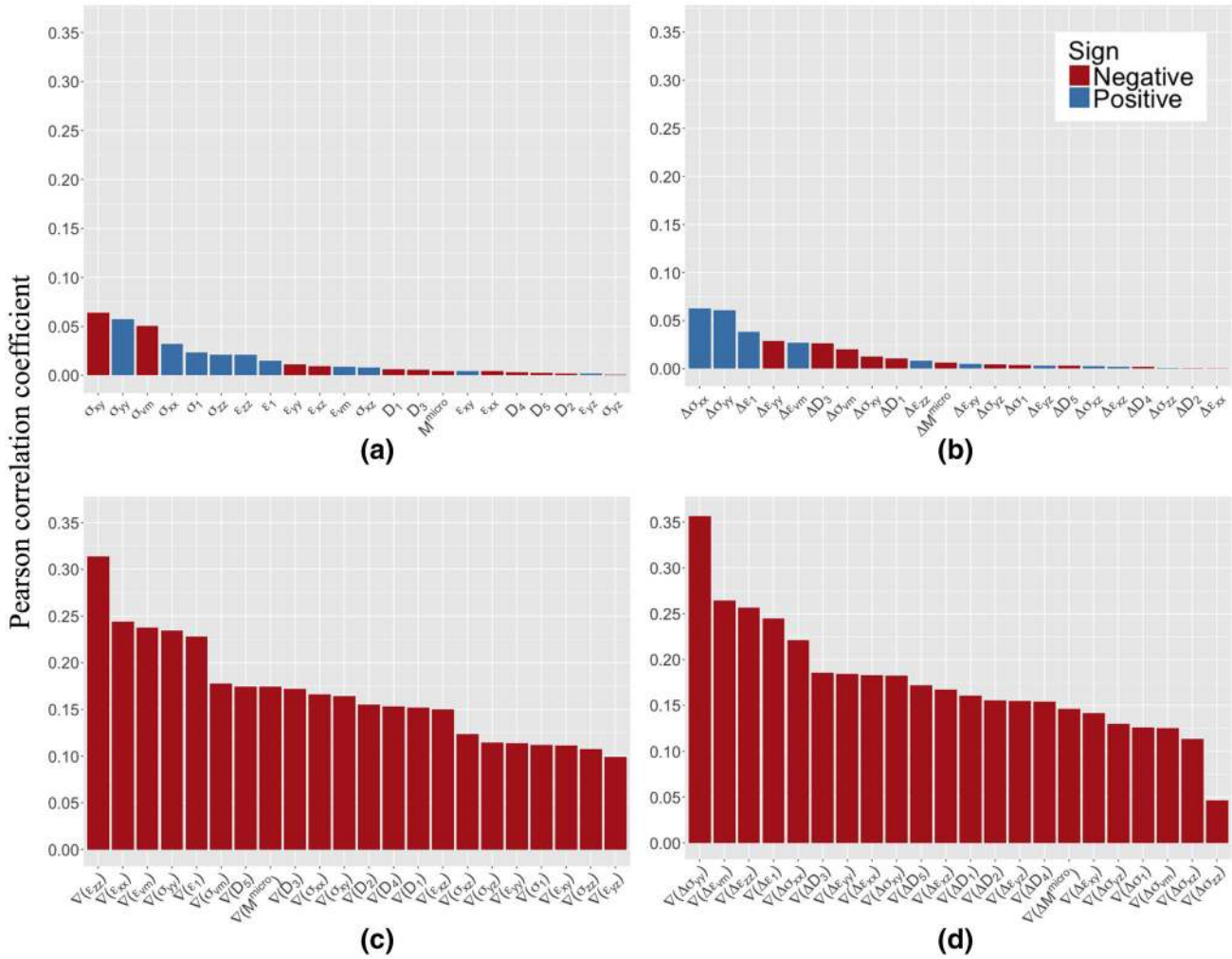

Fig. 2. Correlation coefficients computed between the following metrics and distance to crack surface: (a) field variables, (b) cyclic change in field variables, (c) spatial gradient of field variables, and (d) spatial gradient of cyclic field variables. Correlation coefficients of the spatial gradient values with the known crack surface were found to be consistently stronger than with other, hypothetical crack surfaces. Reprinted with permission from Ref. 30.

small fatigue crack path. Results from the previous correlation analyses are leveraged in the current work involving physics-informed ML to predict the crack surface evolution.

Before describing the methods implemented in this work, it is worth discussing both the limitations and potential benefits of relying on fields from an uncracked microstructure in the ML framework to predict the crack path. The obvious limitation is that the micromechanical fields near a traction-free surface are not the same as fields in the same domain without a traction-free surface. However, the hypothesis explored here is that the micromechanical fields in the uncracked microstructure encode (at least within some finite range of the initiation site) the path that a crack will eventually take. The potential benefit of such approach is that physics-informed ML predictions of crack evolution can be carried out more rapidly than current state-of-the-art approaches for simulating microstructurally small crack evolution. This point is revisited in "Results and Discussion" section.

METHODS

Selection and Extraction of Input Features

Based on the previously described correlation analysis, a new, low-dimensional representation of the data is computed from the original set of features, one that is amenable to spatial relation-based learning algorithms, without losing high-value information contained within the raw features. To achieve this, the existing data are transformed into a new domain that has a low dimensionality for each point in the 3D microstructure. As an analogy, consider how an image is represented in a low-dimensional color space (such as RGB) and contains a vector with three entries at each of the pixels on a two-dimensional (2D) plane.

This representation allows for feasible convolutions and other operations during training. In this work, the dimensionality of each point within the microstructure is first reduced from 88 raw features to a descriptor vector containing three elements, while still retaining the high-value information for predicting the crack path.

The correlation analysis described by Pierson et al.[30] is used to select the subset of features likely to contribute to spatial deviations on the crack surface. This feature selection is imperative to the feasibility of efficient model training; using all 88 features at all 1.7235×10^8 points in the volume would render the computation intractable, given hardware constraints. It was shown in Ref. 30 that the *spatial gradients* of the micromechanical features are more strongly correlated with the crack path than are the micromechanical features themselves, as shown in Fig. 2. Among the 44 spatial gradient features considered, the 22 features associated with the cyclic change between loading and unloading (Fig. 2d) have comparatively higher Pearson correlation coefficients. Thus, the feature set considered for input to the ML models is first down-selected to these 22 features.

Principal component analysis (PCA) is subsequently performed on these 22 features to further reduce the dimensionality of the feature set. PCA is a commonly used dimensionality reduction tool that provides the orthogonal basis vectors in decreasing order of the amount of variance explained along their directions.[34] The 22 features are normalized prior to performing PCA. The PCA analysis shows that the first three modes explain 96% of the total variance, and as such, those three principal components $(\alpha_1, \alpha_2, \alpha_3)$ are selected as the basis for the new data. In other words, the 22 features at each point in the microstructure are projected onto this basis, resulting in a three-element descriptor vector based on the micromechanical metrics at each point.

In addition to the features based on micromechanical fields, an additional feature is considered based purely on the geometrical configuration of the microstructure. Experimental evidence has long suggested that microstructurally small fatigue cracks behave very differently near grain boundaries than within a single grain, manifested as crack deceleration or deflection.[35–37] To account for such a spatially dependent relationship, an additional feature called d_{GB}, which is the distance from a given point to the nearest grain boundary, is added to the descriptor set to produce a location-specific, four-element descriptor. Thus, the descriptor vector $\mathbf{x_i}$ corresponding to each location i can be expressed as follows: $\mathbf{x_i} = [\alpha_{1i}, \alpha_{2i}, \alpha_{3i}, d_{\mathrm{GB},i}]$. For comparison, the ML models are also trained using a descriptor vector of only the experimentally measured Euler angles (represented within the fundamental zone, to account for crystal symmetry) corresponding to each location, to see how predictions based on a simpler, more rudimentary description of the microstructure perform in comparison with the feature vector $\mathbf{x_i}$.

General Approach to Predict Crack Path

The location-specific descriptor vector developed in "Selection and Extraction of Input Features" section is passed on to machine learning (ML) models to predict the z-offset, which represents the vertical deviation of the crack with respect to its neighboring location. This is analogous to predicting the local kink angle along a specified direction. Since the crack surface is assumed to emulate an injective function $f(x, y)$, the problem of predicting the shape of the crack surface is thus reduced to predicting the z-value for a given point (x, y). The ML models are applied using a general prediction approach that is agnostic to the specific ML algorithm used and can be summarized as follows:

1. The local vertical deviation in the crack surface with respect to a "neighboring point," as quantified by the z-offset (Δz), is assumed to be dependent upon a region of influence $\Delta x_v \times \Delta y_v \times \Delta z_v\ \mu\mathrm{m}^3$ centered around that point. In this analysis, we consider $\Delta x_v = \Delta y_v = \Delta z_v = 10\,\mu\mathrm{m}$, meaning that the z-offset at a given point is assumed to be dependent upon microstructural and micromechanical features contained within a cube of dimension $10\,\mu\mathrm{m}$, as shown in Fig. 3. The microstructure is discretized onto a 3D grid with resolution of $1\,\mu\mathrm{m}$,[30] with each point in the grid being defined by the entries of the descriptor vector $\mathbf{x_i}$. The selection of the neighboring point depends on the type of prediction strategy adopted, as explained later in this section.
2. The set of targets (i.e., z-elevation values) paired with the corresponding features is partitioned into training and test sets. The features and the targets in the training set are used to train a given ML algorithm (e.g., CNN model). The trained ML model is used to predict the z-offset, and subsequently compute the z-elevations at test locations using the features in the test set.
3. Optionally, once the predicted z-elevations are obtained, a radial basis function (RBF) smoothing spline is applied on the z-elevations as a postprocessing step to ensure spatial continuity between independently predicted z-offset values. The RBF package in the Python package *scipy* is used to perform this operation. Note that the RBF scheme is only applied in the translational prediction strategy, which is described later in this section. An averaging of multiple predictions is implemented within the radial prediction strategy (also described later), which serves as a postprocessing smoothing operation.
4. Finally, the accuracy of the ML algorithm is evaluated by comparing the predicted (and subsequently corrected using RBF) z-elevations

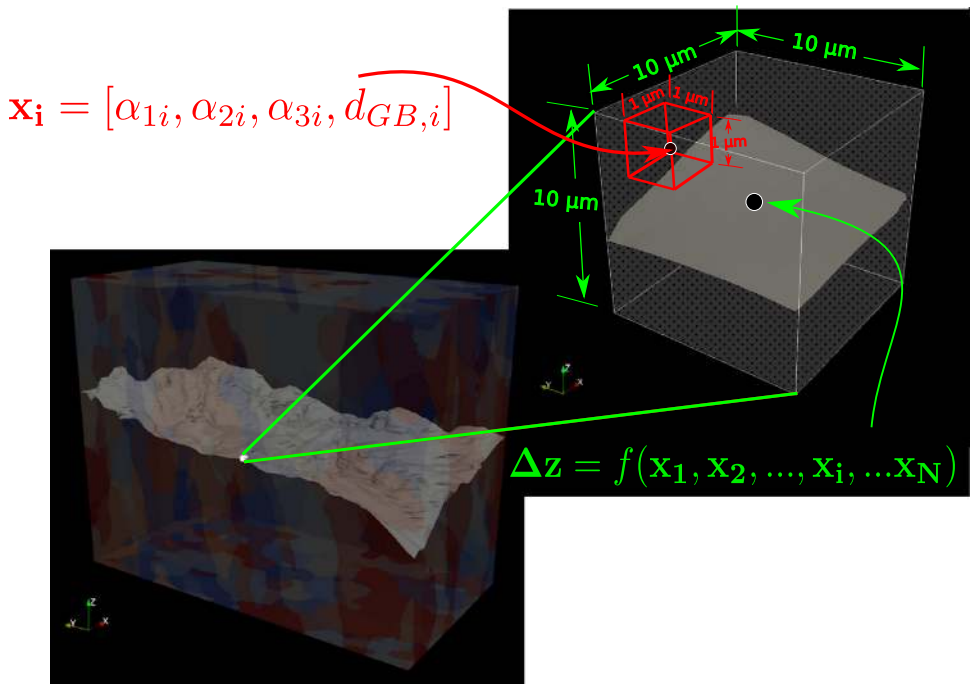

Fig. 3. Schematic of the general approach used in this analysis. A visualization of the measured fatigue failure surface embedded inside the measured microstructural volume $383 \times 750 \times 600\,\mu m^3$ is shown on the left. The microstructural volume is discretized into a grid with $1\,\mu m$ spacing between points, with each point identified using a location-specific descriptor vector $\mathbf{x_i}$. The z-offset (Δz) at a given point is assumed to be a function of all descriptors contained within a volume of $10 \times 10 \times 10\,\mu m^3$ centered around that point, as shown on the right. The goal of machine learning models, therefore, is to find the mapping between Δz and the corresponding set of descriptors.

with the corresponding actual values using the mean squared error and the R^2 metrics.

While the representative dimension of the cubic subvolume considered ($10\,\mu m$) is small relative to the grain size of $\sim O(100\,\mu m)$,[31] note that: (i) a given ML model is used to compute the z-offset at a given point relative to the z-elevation at a neighboring point, ensuring that the local crack surface profile is effectively constrained by descriptor values in neighboring regions (i.e., regions just outside the representative subvolume); and (ii) increasing the dimension of this subvolume would severely increase the number of weight parameters, making the ML model performance prone to loss in accuracy due to overfitting and a high computational cost.

The general strategy presented here is adopted for two prediction approaches based on the way the training and test data are split: (i) translational and (ii) radial. In the translational prediction strategy (Fig. 4), the crack surface is split into two halves, viz. a front half (the half containing the nucleation point of the crack, on which the ML model is trained) and the back half, which is used to evaluate the accuracy of the ML-model predictions. As such, the ML model is trained on all points in the crack surface from 0 to n, where n is the halfway point in the x-direction. During the test phase, the model uses the feature vectors associated with the points at n to predict the z-offsets of the points at $n+1$. Then, the feature vectors associated with the predicted z-positions for the points at $n+1$ are used to predict the z-positions of the points at $n+2$. This continues until the entire crack surface has been predicted. In the translational prediction scheme, 137,445 data points are used to train the CNN model before the trained model is used to predict on 136,710 data points.

In the radial prediction strategy, the ML model is trained on a semicircular region emanating outward from the nucleation point of the crack. The test data (i.e., data for which the predictions are made) corresponds to a region that extends radially outward with respect to the training set, as shown in Fig. 4. This prediction is done by moving pointwise along a semicircular region of the crack surface and predicting the z-offset for three of the adjacent points on the x-y plane. If the predictions of multiple "known" points overlap for a given "unknown" point, the mean of all the predictions is taken. This averaging operation serves as a smoothing operation, as mentioned earlier in this section. Once a prediction for the z-coordinate is produced, the point is added to the "known" region. This continues until the z-coordinates of all points within the domain of the final crack front have been predicted. Figure 4 also presents a schematic of this radial prediction approach. In the radial prediction scheme, 33,985 data points are used to train the CNN model before the trained model is used to predict on 17,754 data points.

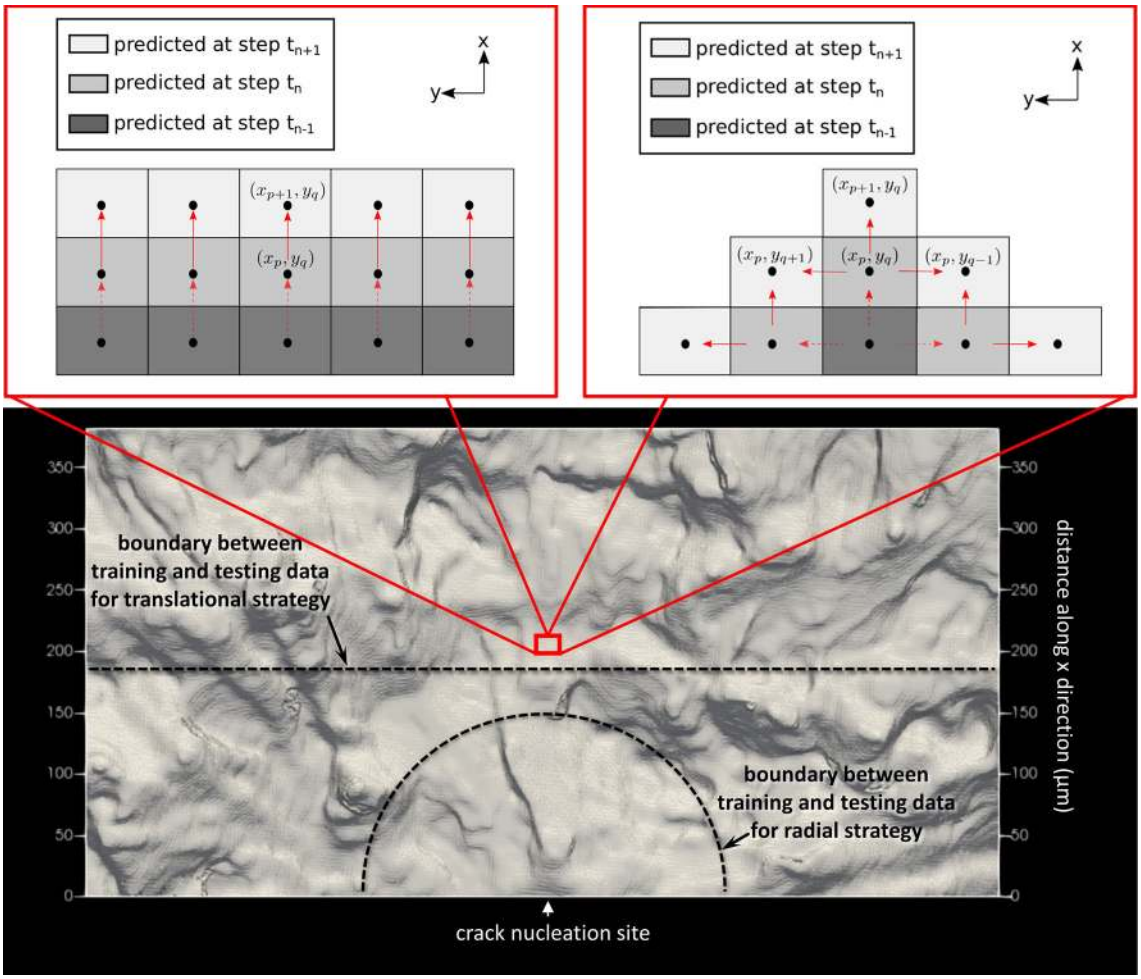

Fig. 4. Two parameterization strategies for predicting the rate-independent evolution of a 3D crack surface. The translational prediction strategy (top left) evolves the crack surface incrementally along the *x* direction. The radial prediction strategy (top right) evolves the crack surface in a nominally radial direction. Dots represent points on the crack surface at a given time step during the evolution procedure. The boundary between training and testing data for each parameterization strategy is shown for reference.

The translational and radial strategies represent two different parameterizations of the crack surface. While cracks evolve physically as arbitrary 3D shapes, expressing and predicting such evolution requires a parametric description (i.e., the definition of a fundamental basis or direction along which the crack path is to be predicted).

Convolutional Neural Network Model to Predict Crack Surface

Convolutional neural networks (CNNs) are considered as the primary ML candidate to predict the crack surface using the general approach presented in "General Approach to Predict Crack Path" section, since CNNs account for the topology of the input data (i.e., allow extraction of higher-level features by considering local correlations among spatially proximal lower-level features[38]). CNNs usually employ a hierarchical structure to determine how these local feature maps correlate with the response variable of interest. A CNN model is defined by weight tensors that indicate the relative influence that a spatial arrangement of features has on a given output. Aside from ensuring a lower number of free weight parameters, the spatial sharing of the weights also enforces a degree of shift invariance within the model.[38] Through an optimization process, the CNN model "learns" how the spatial arrangement of the specific features is related to the output variable to be predicted. Once trained, the CNN model can then be used in a computationally efficient, forward sense to make predictions.

The application of CNNs is pertinent to the problem of predicting the *z*-offset presented here, as the *z*-offset at a given point is likely to be dependent on the relative spatial arrangement of the descriptors (as developed in "Selection and Extraction of Input Features" section) within a given region—in our case, a cubic volume surrounding that point. It is thus hypothesized that the proposed CNN model would perform comparatively better than other ML algorithms. XGBoost, a

scalable tree boosting algorithm that has achieved state-of-the-art results in several ML challenges[39] and has been previously employed in materials science-related applications,[40] is selected as the primary ML model for comparison with the proposed CNN. Note that support vector regression (SVR) was also initially considered as an ML candidate; however, SVR exhibited significantly poor performance (e.g., $R^2 < 0.6$ for the radial strategy), aside from suffering from poor scalability of computational time with increasing size of training data.[41]

Figure 5 presents a schematic of the CNN architecture implemented to predict the crack surface, as quantified by the z-offsets. For any given point on the crack surface, we consider as input a $10\,\mu m \times 10\,\mu m \times 10\,\mu m$ volume surrounding the point, where each voxel within the input volume occupies $1\ \mu m^3$ and has associated with it the feature vector $\mathbf{x_i}$ (or an input vector describing the Euler angles). A filter (which can be thought of as a scanning volume) of size $3\,\mu m \times 3\,\mu m \times 3\,\mu m$ then convolves (or scans) through the input volume. Each position within the scanning volume has an initially arbitrary coefficient, or weight value, associated with it, which, considering the entire scan volume, can be represented by a weight tensor, $\bar{\omega}$. This weight tensor for a given filter is initialized at the beginning of the training phase. As the scanning volume convolves throughout the larger input volume, each element in the feature vector $\mathbf{x_i}$ at a given point in the scanning volume is multiplied by the corresponding weight value at that location. This convolution process continues until the scanning volume has scanned the entire input volume. The output from the convolution step is a matrix of weighted values representing the spatial arrangements of the $\mathbf{x_i}$ features. The next step shown in Fig. 5 is to apply a nonlinear transfer function to the elements of the matrix, whose effect (over subsequent iterations, or epochs) is to emphasize important features and deemphasize nonimportant features. In this work, we apply a rectified linear (or ReLu) transfer function. Once the matrix is updated to reflect the application of the nonlinear transfer function, a process known as "max pooling" is carried out, in which the updated matrix is subdivided into regions that correspond to neighborhoods from the original input volume. The maximum value from each subregion of the matrix is extracted and input to a new, lower-dimensional matrix that represents a new set of intermediate features for the volume (rather than the original set of $\mathbf{x_i}$ features). All of the above steps are carried out using multiple independent filters. In this work, four filters are applied, which results in four new, lower-dimensional feature matrices.

As shown in Fig. 5, the process of convolving is repeated after max pooling, this time using the resulting, lower-dimensional, feature matrices just described. As before, a nonlinear transfer function, followed by max pooling, is applied after the convolution. The result now is an even lower-dimensional, but high-value, representation of the spatial arrangements of the original features. The matrix is then flattened and mapped to the output (z-offset) via a multilayer perceptron (i.e., a conventional artificial neural network). A technique called "dropout" is also applied, which is a regularization technique used in deep neural networks that prevents overfitting. In this paper, the dropout operation is also used to predict the uncertainty associated with the CNN predictions, which is described later in "Computing Model Uncertainty Associated with CNN Predictions" section. The error in the predicted z-offset at the given training point on the crack surface is then backpropagated to update the weight values (coefficients) in the CNN model. During training, the weight tensors are optimized through gradient descent and backpropagation using ADAM,[42] a stochastic optimization algorithm. The CNN model is thus defined by the weight tensors in the convolution and the fully connected layer. Once the model is optimized, it can be used in a forward prediction with significant computational efficiency.

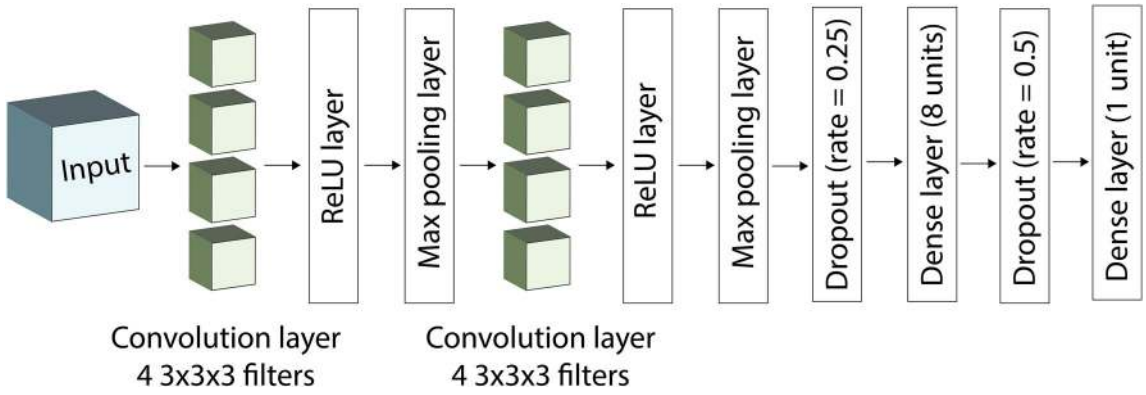

Fig. 5. Architecture of the convolutional neural network used in this work.

XGBoost as a Comparative ML Model

The performance of the CNN is compared against that of XGBoost, a type of gradient-boosting algorithm[39] that does not take into account the spatial arrangements of features. Gradient-boosting algorithms make predictions by using an ensemble of predictive ML models, where these predictive models are added sequentially over multiple iterations to minimize a given loss function.[43] In the case of gradient tree boosting algorithms, the predictive ML model takes the form of a decision tree, in which first- and second-order derivatives of the loss function with respect to the function value (i.e., prediction of each decision tree architecture) are used to determine the optimal decision tree architecture.

XGBoost is a variant of gradient tree boosting algorithm with multiple algorithm modifications and parallelization techniques to improve its computational efficiency, the details of which are available in other literature.[39] The CNN is hypothesized to perform better than XGBoost for two main reasons: (i) the CNN inherently accounts for the spatial arrangement of local features within the microstructure (as represented by the descriptor vector or by the set of Euler angles), and (ii) the CNN model might be better suited to predicting a continuous variable such as z-offset compared with XGBoost, which constructs a piecewise-constant model for each leaf in the decision tree. This hypothesis is explored in "Results and Discussion" section.

Computing Model Uncertainty Associated with CNN Predictions

Uncertainty quantification of parameters pertaining to fatigue phenomena is critical for reliability analysis and safety evaluation of structural components.[44,45] The sources of such uncertainties can be diverse, such as variability in material properties, data uncertainty, and model uncertainty.[46] In our study, quantification of model uncertainty is most relevant.[46] Previous work by Rovinelli et al.[7] has successfully explored Bayesian approaches for estimating the uncertainties associated with predictions of global measures (viz., residual life); however, it is of interest in this work to quantify the uncertainties associated with estimation of local parameters relating to small cracks. One of the contributions of this paper, therefore, is to demonstrate how we can leverage the CNN model presented in "Convolutional Neural Network Model to Predict Crack Surface" section to compute the model uncertainty associated with the predictions of local z-elevation at given location (x, y), based on microstructural and micromechanical features in proximity to that location.

As such, the dropout operation mentioned in "Convolutional Neural Network Model to Predict Crack Surface" section is utilized to quantify the uncertainty in local z-elevation prediction. As mentioned in "Convolutional Neural Network Model to Predict Crack Surface" section, the dropout layer can be used to stochastically regularize the CNN model against overfitting by aggregating over multiple neural network configurations.[47] When dropout is applied between two layers l and $l+1$ (see Fig. 5), the interconnection between a given intermediate feature (also known as a "node" in the multilayer perceptron) in layer l and another intermediate feature in layer $l+1$ is retained with probability p. This means that, in a given single iteration during training, the set of intermediate features in layer $l+1$ is obtained by means of randomly subsampling from the set of intermediate features in the previous layer l with probability p.

Gal and Ghahramani[48] suggested that, beyond simply preventing overfitting during training, the dropout layer can be further exploited during prediction to compute the uncertainty of a deep neural network model prediction, without needing any modifications to the model or the optimization objective function, and without compromising the prediction accuracy. They showed that the loss function for a neural network model with dropout is equivalent to a loss function that minimizes the Kullback–Leibler divergence between an approximate distribution and the posterior distribution of a deep Gaussian process.[48,49] The approximate distribution of predictions in this case is obtained through multiple samplings of the z-offset using the CNN model with dropout. As the objective here is to obtain a distribution of predictions, the training configurations are retained for inference as well; i.e., rather than rescaling the predictions by p, the interconnections between nodes are each dropped with probability p during prediction, just as was done during the training phase.

In this analysis, therefore, we leverage this finding from Gal and Ghahramani to not only predict the crack path, but also to compute the uncertainty associated with the CNN predictions of the crack path at a given location. The prediction of the crack path at a given point can depend on prior predictions at neighboring points, and as such, the associated uncertainty propagation is accounted for using the uncertainty propagation rules, as presented in Ref. 50.

RESULTS AND DISCUSSION

Table II presents the accuracies of the 3D crack path predictions using CNN and XGBoost, as obtained using the translational prediction strategy shown in Fig. 4. To indicate the relative benefits of using these ML algorithms, the accuracies obtained are compared with a baseline hypothetical crack surface, which is obtained by simply extruding the crack surface in the x-direction at the end of the training region through the entire prediction region. This hypothetical crack surface is treated as the naive prediction. The improvements in the R^2 value

Table II. Prediction results of crack surface elevation for different combinations of ML model and input feature type

Model	Feature type	R^2	RMSE (μm)	% Increase in R^2
Naive approach	N/A	0.788	14.17	–
CNN (with RBF)	Descriptor vector, $\mathbf{x_i}$	0.891	10.16	+13.1
CNN (with RBF)	Euler angles	0.844	12.16	+7.1
CNN (without RBF)	Descriptor vector, $\mathbf{x_i}$	0.832	12.60	+5.73
XGBoost	Descriptor vector, $\mathbf{x_i}$	0.888	10.31	+12.6
XGBoost	Euler angles	0.669	17.69	−15.2

relative to the naive prediction are presented in the final column of Table II. All of the models, with the exception of XGBoost trained using only the Euler angles, lead to better predictions of 3D crack path than simply taking the naive approach of extending the crack forward. Overall, the CNN with RBF smoothing outperforms the XGBoost model for a given training set.

The results in Table II show that both the CNN (with RBF smoothing) and XGBoost models exhibit improvement in accuracy when the descriptor vector (i.e., PCA values and d_{GB}) developed in "Selection and Extraction of Input Features" section is used as the feature type, compared with cases where only Euler angles are used. However, note that the improvement in accuracy is significantly higher in the case of XGBoost (improvement in R^2 from 0.669 to 0.88), compared with the relatively marginal improvement when CNN is selected as the ML algorithm (improvement in R^2 from 0.844 to 0.891). While this observation supports our initial hypothesis that the encoded descriptor vector $\mathbf{x_i}$ can serve adequately as a local reduced-order representation of the microstructural and micromechanical features, it also indicates that, in comparison with the XGBoost model, the CNN gains less relative benefit from having access to the high-fidelity, finite-element-based inputs than simply having access to the crystal orientations at each point in the discretized domain. One possible reason for this is that the CNN's ability to account for the spatial arrangement of crystal orientations allows it to learn the relevant relationships that would otherwise be learned from the finite-element results. For a non-spatially-aware ML model, such as XGBoost, it is imperative that the training dataset accounts, *a priori*, for the spatial relationships of crystal orientations, whereas this is not necessarily required for the CNN model. In the future, considering alternative features that represent crystal orientations more uniquely than Euler angles (see work by Mangal and Holm[25]) could help to generalize the CNN's predictive capability across microstructural datasets.

In addition to the quantitative error measures provided in Table II, Fig. 6 provides a visualization of the crack surfaces predicted using the different ML models with different input feature types under the translational prediction strategy. Figure 6c and f correspond to the predictions made using the CNN (with RBF) and XGBoost, respectively, both using the

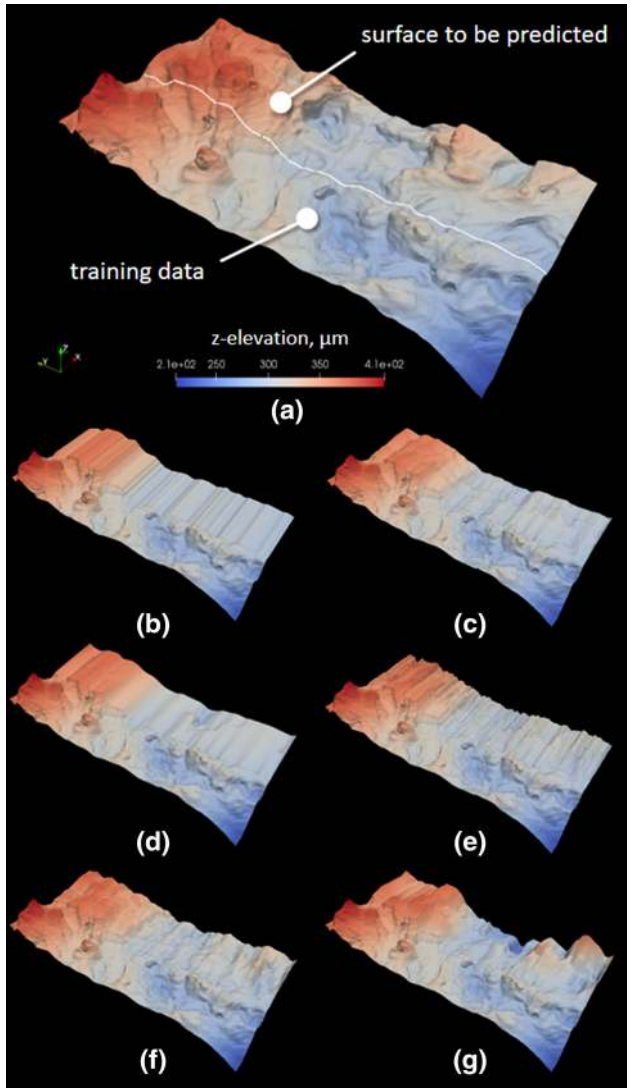

Fig. 6. 3D crack surface obtained using the translational parameterization strategy for predicting crack path: (a) actual crack surface, (b) naive prediction, (c) CNN prediction using proposed descriptor vector ($\mathbf{x_i}$) and RBF smoothing, (d) CNN prediction using Euler angles and RBF smoothing, (e) CNN prediction using proposed descriptor vector ($\mathbf{x_i}$) and without RBF smoothing, (f) XGBoost prediction using proposed descriptor vector ($\mathbf{x_i}$), and (g) XGBoost prediction using Euler angles.

descriptor vector $\mathbf{x_i}$. These two predictions correspond to a 13.1% and 12.6% improvement, respectively, in R^2 compared with the naive prediction, representing the greatest relative improvements among all of the approaches considered. Interestingly, Fig. 6g shows that, when XGBoost is used with Euler angles for training and prediction, highly anomalous behavior is predicted in the crack surface. While the resulting crack surface looks, qualitatively, very realistic, it in fact has the lowest performance (a 15.2% relative reduction in R^2 compared with simply taking the naive approach to predict the crack surface).

The cumulative uncertainties associated with the crack path predictions using the translational parameterization strategy are shown in Fig. 7. In Fig. 7, three arbitrary sections (AA′, BB′, and CC′) are shown to demonstrate the actual versus predicted crack paths along the nominal crack growth direction. The predictions correspond to the CNN model, trained with the descriptor vector $\mathbf{x_i}$ (without RBF smoothing, to isolate the uncertainty associated with the CNN model). For each section, the entire crack path is shown, including the portion used for training. The portion used for blind prediction by the CNN model is magnified, and the predicted crack path is colored by the cumulative uncertainty associated with the CNN model's prediction. Note that the cumulative uncertainty in the x-direction is computed using the pointwise uncertainty that is associated with only the CNN model and does not account for the uncertainties in inputs or in modeling choices made prior to the application of the CNN. While the CNN predictions generally capture the trends of the crack path, there are obvious deviations between the predicted and actual crack paths in certain regions. The uncertainty values, however, do not track directly with these deviations, since uncertainty in the model predictions is independent of the known crack surface. Generally, the CNN model uncertainty is higher in regions where the model predicts a steep change in the z-offset value. This is visualized in Fig. 8, which shows two maps from a top-down view of the prediction region. One map shows the pointwise (noncumulative) uncertainty, while the other shows the corresponding prediction of the z-offset by the CNN model. Visually, there is a clear correspondence between the regions where a steep change in crack path is predicted (either positive or negative) and where the model estimates a relatively large uncertainty value. By presenting the model with

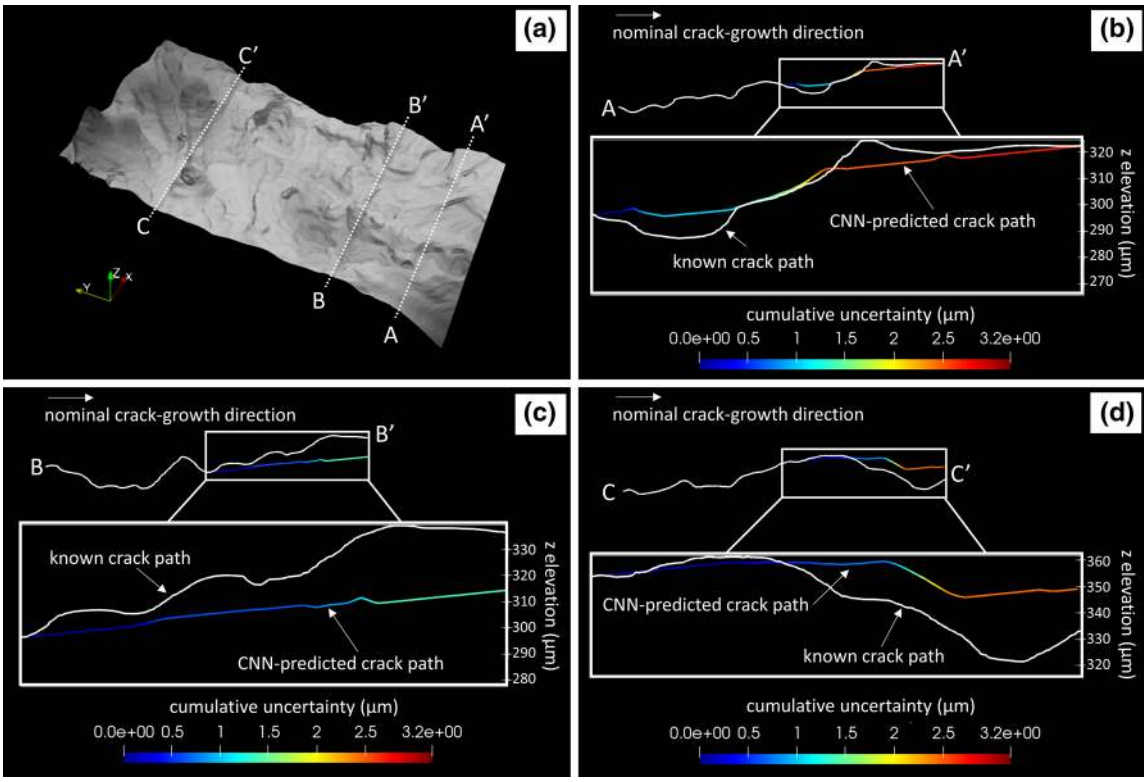

Fig. 7. Comparison of measured versus CNN-predicted crack path along with cumulative uncertainty for the translational parameterization strategy (crack path predicted along the x direction): (a) location of three arbitrary reference sections (AA′, BB′, and CC′) depicted on a rendering of the actual crack surface; entire crack path along (b) AA′, (c) BB′, and (d) CC′, with a magnified view of the region containing the prediction. The predicted path is colored based on the cumulative uncertainty associated with the CNN model. For interpretation of color, please see online version.

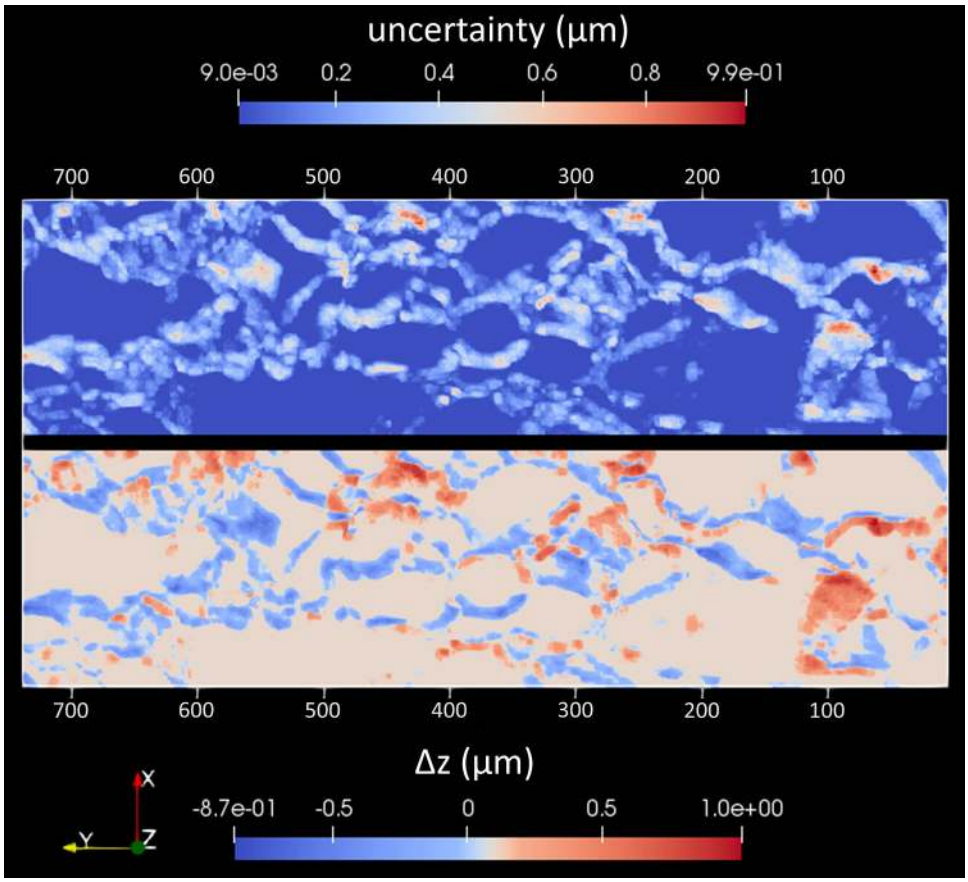

Fig. 8. Top-down view of prediction region for the translational parameterization strategy (crack growth is nominally in the *x* direction). Map of pointwise (noncumulative) uncertainties associated with the CNN predictions of *z*-offset (Δz) along with a map of corresponding Δz values predicted by the CNN (trained with descriptor vector $\mathbf{x_i}$ and no RBF smoothing). The *y*-axis scale is shown in μm.

Table III. Crack surface height prediction results for different ML models using the proposed descriptor vector (radial prediction strategy)

Model	Feature type	R^2	RMSE (μm)
CNN	Descriptor vector, $\mathbf{x_i}$	0.842	8.410
XGBoost	Descriptor vector, $\mathbf{x_i}$	0.784	9.344

more training data that include instances of steep elevation changes, the uncertainty in such predictions is expected to decrease.

The ML algorithms are now compared for the radial prediction strategy, focusing on the CNN and XGBoost models with the descriptor vector, $\mathbf{x_i}$, used as input. Table III presents the accuracies of the ML algorithms corresponding to the radial prediction strategy, which is more likely to resemble crack front profiles than the translational strategy. Figure 9a presents the actual crack surface, with the boundary indicating the split between the training and test regions. Note that the outer profile of the predicted region is not perfectly semicircular. This is because the predictions were truncated at the last known crack front profile from experiment.[31] The comparison in Table III shows that, for the radial prediction strategy, CNN performs marginally better than XGBoost. Inspection of Fig. 9b, c and comparison with 9a also reveals that CNN performs noticeably better than XGBoost in regions where the crack surface elevation is abnormally high (i.e., z-elevation $\geq 340\ \mu m$). Hence, note that, while global measures such as R^2 and RMSE provide valuable, quantitative, information for an aggregated measure of accuracy of a given ML model, these measures are homogenized over the entire crack surface domain and do not necessarily convey information about how ML algorithms predict the crack surface in these anomalous, local regions. A qualitative comparison of the ML-predicted crack surface with the actual crack surface—particularly in such anomalous regions—can, therefore, provide additional insight into the performance of the ML model in predicting the crack path.

As stated in "General Approach to Predict Crack Path" section, the size of the training data corresponding to the radial prediction strategy is less than 25% of that corresponding to the translational case, so it is possible that the CNN is able to generalize comparatively better than XGBoost when fewer training data are available. One potential reason for the improved generalizability of the

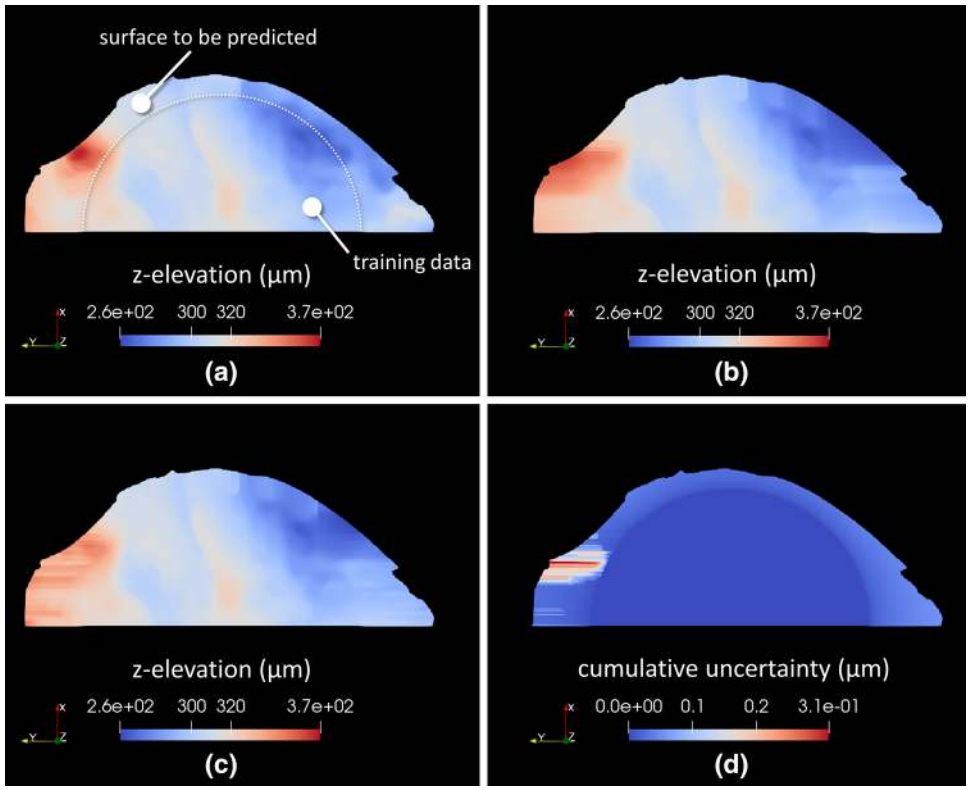

Fig. 9. 3D crack surface obtained using the radial parameterization strategy for predicting crack path: (a) actual crack surface, (b) CNN prediction using proposed descriptor vector ($\mathbf{x}_i$) and RBF smoothing, (c) XGBoost prediction using proposed descriptor vector ($\mathbf{x}_i$), and (d) cumulative uncertainty associated with CNN predictions. Each surface has been cropped to the shape of the final crack-front profile measured in experiment presented in Ref. 31.

CNN could be due to its ability to account for the spatial distribution of descriptors, as discussed in "Convolutional Neural Network Model to Predict Crack Surface" section—for which XGBoost does not inherently account. In the context of a regression problem, XGBoost, which is a decision tree-based algorithm, constructs a piecewise-constant model with each leaf fit to a single value during training. When the size of the training data is comparatively smaller, it is possible that the leaf values are fit to a discrete value (corresponding to a z-offset) that is not sufficiently resolved.

Figure 9d shows the cumulative uncertainty maps corresponding to the predictions presented in Fig. 9b using the method presented in "Computing Model Uncertainty Associated with CNN Predictions" section. The figure shows that the uncertainties in CNN predictions are comparatively high in regions where the z-elevations are high, which could be due to the relative lack of data points with high z-elevations in the training region, as mentioned previously.

The implementation of the ML framework presented here is subject to several limitations. One of the limitations of the presented work is that only one set of experimental data of the crack surface, along with the corresponding raw microstructural and micromechanical features from the uncracked volume, is available to train the ML models. While the current crack surface does exhibit anomalous regions in crack path deviation, the same framework is anticipated to be applicable to (and perhaps even more useful for) more tortuous crack surfaces. In that regard, there is still a need to investigate the performance of the proposed ML framework on similar high-resolution datasets beyond the one presented here. The proposed framework also does not explicitly handle crack initiation, but instead predicts the crack surface path once the initiation site is known. Finally, as mentioned at the end of "Prior Work by the Authors" section, one could view the lack of an explicit discontinuity in the training data as a limitation of the framework. However, the hypothesis explored here is that the micromechanical fields in the uncracked microstructure encode (within some spatial range of the initiation site) the path that a crack will eventually take. It is expected that, beyond this finite range, the fields will hold less predictive information, at which point an actual crack surface might need to be modeled. In fact, it is anticipated that a hybrid of the ML framework could be integrated with a computationally expensive, high-fidelity model of an actual evolving discontinuity to help advance the crack in a more computationally efficient manner. This is an area for future investigation.

The proposed framework for predicting crack paths could prove useful in various materials science applications. For example, the same method for predicting the crack surface elevation could be implemented for predicting the local crack growth rate, which could lead to improved structural-prognosis capabilities. In addition to structural prognosis, the ML framework for predicting crack paths could also be applied in materials design applications. For example, rapid predictions of crack paths could be made for various microstructural instantiations under a given set of boundary conditions to identify or downselect optimal microstructural arrangements for controlled crack propagation.

In summary, the proposed framework presents an approach by which relevant features, i.e., microstructural features and simulated (computed) micromechanical fields under cyclic loading using the *uncracked* material, can be used to predict the evolution of a 3D crack path in a polycrystalline material. The promising predictive capability of the ML framework indicates that one can potentially predict the evolution of the crack surface (with some prior knowledge of crack initiation), without modeling the explicit discontinuity inside the high-fidelity model, and encourages future research to potentially address the existing limitations mentioned above. Based on the presented work, the scope for future research includes, but is not limited to: (i) development of new benchmark datasets for investigating the performance of the proposed ML framework (or that of any other predictive models); (ii) analysis of the CNN model by adapting/modifying existing methods for model interpretation of deep neural networks, with the aim of gaining insight into the factors and mechanisms that govern small crack growth; and (iii) development of methodologies that allow coupling/information flow between the CNN model and existing techniques for computational fracture mechanics.

CONCLUSION

A methodology is presented to leverage microstructural and micromechanical data from an uncracked polycrystal to train a machine learning (ML) model to predict a 3D fatigue crack path (assuming some prior knowledge of crack initiation). The framework presented includes a data parameterization strategy, ML model setup, uncertainty quantification associated with ML model predictions, and ML model assessment. The data used as a proof of concept are derived from previous experimental measurements and high-fidelity, crystal-plasticity modeling of an Al-Mg-Si alloy. The high-dimensional data are reduced to a low-dimensional, high-value representation and subsequently passed to a convolutional neural network (CNN). A portion of the crack surface is used for training, while the remaining portion is predicted using the ML model. A number of comparisons are made regarding the prediction performance, including the effect of using a radial basis function (RBF) to ensure spatial continuity of the independent predictions of crack-surface elevation and the effect of using only Euler angles as the input feature rather than the low-dimensional feature vector that includes micromechanical information. The predictions from the CNN model are also compared with predictions from XGBoost, a scalable tree-boosting algorithm that has recently been used in materials science-related applications. The following key conclusions can be drawn from this analysis:

- The proposed descriptor vector, which is used as a reduced-order representation of local micromechanical and microstructural features, performs better (in terms of R^2 and RMSE) than Euler angles in predicting the 3D crack surface when used as the feature set in a given ML algorithm. However, the improvement in accuracy is only significant for the XGBoost algorithm. In the case of CNN, the relative advantage in using the descriptor vector is marginal.
- In the absence of micromechanical fields, i.e., with only crystal orientations as inputs, the CNN performs fairly well ($R^2 = 0.844$) in predicting the crack path. A likely reason for this is that the CNN's ability to account for the spatial arrangement of crystal orientations allows it to learn the relevant relationships that would otherwise be learned from finite-element simulations.
- The CNN model also allows for computation of model uncertainties associated with its predictions by using the dropout operation, which can be used to compute (and visualize) the uncertainty maps corresponding to the predicted crack surface.

Once trained, the ML model could offer a powerful approach to predicting, relatively rapidly, the expected crack evolution through an otherwise uncracked microstructure (within some limited domain). More experimental observations would enhance the training data and facilitate model generalization.

ACKNOWLEDGEMENTS

This material is based on research sponsored by the Air Force Office of Scientific Research Young Investigator Program under Agreement No. FA9550-15-1-0172. The support and resources of the Center for High Performance Computing at the University of Utah are gratefully acknowledged.

REFERENCES

1. H. Mughrabi, *Phil. Trans. R. Soc. A* **373**(2038), 20140132 (2015).
2. S. Kumar and W.A. Curtin, *Mater. Today* **10**(9), 34 (2007).
3. P. Paris and F. Erdogan, *J. Basic Eng.* **85**(4), 528 (1963).

4. D. Davidson, K. Chan, R. McClung and S. Hudak, *Comprehensive Structural Integrity: Small Fatigue Cracks*, ed. I. Milne, R.O. Ritchie, and B.L. Karihaloo (Elsevier, 2003), pp. 129–164.
5. K. Hussain, *Eng. Fract. Mech.* **58**(4), 327 (1997).
6. A.J. McEvily, *J. Soc. Mater. Sci.* **47**(3Appendix), 3 (1998).
7. A. Rovinelli, Y. Guilhem, H. Proudhon, R.A. Lebensohn, W. Ludwig, and M.D. Sangid, *Modell. Simul. Mater. Sci. Eng.* **25**(4), 045010 (2017).
8. A. Fatemi and L. Yang, *Int. J. Fatigue* **20**(1), 9 (1998).
9. J. Bozek, J. Hochhalter, M. Veilleux, M. Liu, G. Heber, S. Sintay, A. Rollett, D. Littlewood, A. Maniatty, and H. Weiland et al., *Modell. Simul. Mater. Sci. Eng.* **16**(6), 065007 (2008).
10. M. Li, S. Ghosh, O. Richmond, H. Weiland, and T. Rouns, *Mater. Sci. Eng. A* **265**(1–2), 153 (1999).
11. R.J. Asaro, *J. Appl. Mech.* **50**(4b), 921 (1983).
12. G. Potirniche, S. Daniewicz, and J. Newman Jr., *Fatigue Fract. Eng. Mater. Struct.* **27**(1), 59 (2004).
13. S. Suresh and R. Ritchie, *Int. Metals Rev.* **29**(1), 445 (1984).
14. W.D. Musinski and D.L. McDowell, *Int. J. Fatigue* **37**, 41 (2012).
15. C.P. Przybyla, W.D. Musinski, G.M. Castelluccio, and D.L. McDowell, *Int. J. Fatigue* **57**, 9 (2013).
16. A. Agrawal and A. Choudhary, *APL Mater.* **4**(5), 053208 (2016).
17. Y. Liu, T. Zhao, W. Ju, and S. Shi, *J. Mater.* **3**(3), 159 (2017).
18. A.D. Spear, S.R. Kalidindi, B. Meredig, A. Kontsos, and J.B. le Graverend, *JOM* **70**, 1143 (2018).
19. M.W. Gaultois, A.O. Oliynyk, A. Mar, T.D. Sparks, G.J. Mulholland, and B. Meredig, arXiv preprint arXiv:1502.07635 (2015).
20. J. Ling, M. Hutchinson, E. Antono, B. DeCost, E.A. Holm, and B. Meredig, *Mater. Discov.* **10**, 19 (2017).
21. B.L. DeCost and E.A. Holm, *Comput. Mater. Sci.* **110**, 126 (2015).
22. J. Xu, X. Luo, G. Wang, H. Gilmore, and A. Madabhushi, *Neurocomputing* **191**, 214 (2016).
23. S. Jha, R. Brockman, R. Hoffman, V. Sinha, A. Pilchak, W. Porter, D. Buchanan, J. Larsen, and R. John, *JOM* **70**, 1147 (2018).
24. N.H. Paulson, M.W. Priddy, D.L. McDowell, and S.R. Kalidindi, *Int. J. Fatigue* **119**, 1 (2019).
25. A. Mangal and E.A. Holm, *Int. J. Plast.* **111**, 122 (2018). https://doi.org/10.1016/j.ijplas.2018.07.013.
26. A. Rovinelli, M.D. Sangid, H. Proudhon, and W. Ludwig, *npj Comput. Mater.* **4**(1), 35 (2018).
27. H. Fujii, M. DJC, and B. HKDH, *ISIJ Int.* **36**(11), 1373 (1996).
28. A. Cecen, H. Dai, Y.C. Yabansu, S.R. Kalidindi, and L. Song, *Acta Mater.* **146**, 76 (2018).
29. Z. Yang, Y.C. Yabansu, R. Al-Bahrani, W.K. Liao, A.N. Choudhary, S.R. Kalidindi, and A. Agrawal, *Comput. Mater. Sci.* **151**, 278 (2018).
30. K.D. Pierson, J.D. Hochhalter, and A.D. Spear, *JOM* **70**(7), 1159 (2018). https://doi.org/10.1007/s11837-018-2884-2.
31. A.D. Spear, S.F. Li, J.F. Lind, R.M. Suter, and A.R. Ingraffea, *Acta Mater.* **76**, 413 (2014).
32. A.D. Spear, J.D. Hochhalter, A.R. Cerrone, S.F. Li, J.F. Lind, R.M. Suter, and A.R. Ingraffea, *Fatigue Fract. Eng. Mater. Struct.* **39**(6), 737 (2016).
33. D. Raabe, M. Sachtleber, Z. Zhao, F. Roters, and S. Zaefferer, *Acta Mater.* **49**(17), 3433 (2001). https://doi.org/10.1016/s1359-6454(01)00242-7.
34. I. Jolliffe, in *International Encyclopedia of Statistical Science* (Springer, Berlin, 2011), p. 1094.
35. J. Lankford, *Fatigue Fract. Eng. Mater. Struct.* **5**(3), 233 (1982). https://doi.org/10.1111/j.1460-2695.1982.tb01251.x.
36. T.G. Zhai, A. Wilkinson, and J. Martin, *Acta Mater.* **48**(20), 4917 (2000).
37. K. Miller, *Inst. Mech. Eng. Part C Mech. Eng. Sci.* **205**(5), 291 (1991).
38. Y. LeCun and Y. Bengio et al., *Handb. Brain Theory Neural Netw.* **3361**(10), 1995 (1995).
39. T. Chen and C. Guestrin, in *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2016), p. 785.
40. L. Laugier, D. Bash, J. Recatala, H.K. Ng, S. Ramasamy, C.S. Foo, V.R. Chandrasekhar, and K. Hippalgaonkar, arXiv preprint arXiv:1811.06219 (2018).
41. A. Bordes, S. Ertekin, J. Weston, and L. Bottou, *J. Mach. Learn. Res.* **6**(Sep), 1579 (2005).
42. D.P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).
43. J.H. Friedman, *Ann. Stat.* **29**, 1189 (2001).
44. M.S. Chowdhury, C. Song, and W. Gao, *Eng. Fract. Mech.* **78**(12), 2369 (2011).
45. S.C. Kang, H.M. Koh, and J.F. Choo, *Probab. Eng. Mech.***25**(4), 365 (2010).
46. S. Sankararaman, Y. Ling, and S. Mahadevan, *Eng. Fract. Mech.* **78**(7), 1487 (2011).
47. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *J. Mach. Learn. Res.* **15**(1), 1929 (2014).
48. Y. Gal and Z. Ghahramani, in *International Conference on Machine Learning*, p. 1050 (2016).
49. Y. Gal and Z. Ghahramani, arXiv preprint arXiv:1506.02158 (2015).
50. A summary of error propagation (2007). http://ipl.physics.harvard.edu/wp-uploads/2013/03/PS3_Error_Propagation_sp13.pdf. Accessed 14 June 2019.