# Predictive modeling of dynamic fracture growth in brittle materials with machine learning

Moore, Bryan Alexander
Srinivasan, Gowri
O'Malley, Daniel
Viswanathan, Hari S.
Hunter, Abigail
Rougier, Esteban



**To be published in:** Computational Materials Science

**DOI to publisher's version:** 10.1016/j.commatsci.2018.01.056

# Predictive modeling of dynamic fracture growth in brittle materials with machine learning

Bryan Moore, Esteban Rougier, Daniel O'Malley, Gowri Srinivasan, Abigail Hunter, Hari Viswanathan

**Abstract**

We use simulation data from a high fidelity Finite-Discrete Element Model to build an efficient Machine Learning (ML) approach to predict fracture growth and coalescence. Our goal is for the ML approach to be used as an emulator in place of the computationally intensive high fidelity models in an uncertainty quantification framework where thousands of forward runs are required. The failure of materials with various fracture configurations (size, orientation and the number of initial cracks) are explored and used as data to train our ML model. This novel approach has shown promise in predicting spatial (path to failure) and temporal (time to failure) aspects of brittle material failure. Predictions of where dominant fracture paths formed within a material were ∼85% accurate and the time of material failure deviated from the actual failure time by an average of ∼16%. Additionally, the ML model achieves a reduction in computational cost by multiple orders of magnitude.

*Keywords:*
Machine Learning; Finite Discrete Element Models; Brittle Fracture Networks; Multi-scale; Neural Networks

## 1. Introduction

Failure in brittle materials occurs through the propagation of fractures and has been investigated for over a century due to its many applications in industry (aviation systems [1], subsurface fracturing [2], etc.) and inherent scientific curiosity [3, 4, 5]. Brittle materials are materials in which cracks of atomic sharpness essentially propagate by bond ruptures [6]. Unlike ductile failure behavior, brittle solids fail with little warning along grain boundaries [3] or in the case of geomaterials, fractures propagate along interstitials [7]. Once fracturing has begun, the fractures will have a strong tendency to propagate since they are driven by internally stored elastic energy. Accurately predicting failure at the macroscale requires knowledge of the dynamically evolving microstructure, including features such as grain boundaries and interstitials, and also accounting for fracture interactions with the microstructure and other fractures present in the system. Many approaches have been taken to address the failure of a material at different length scales including analytical and numerical methods [8, 9, 10, 11, 12, 13, 4]. Modeling micro- and meso-scale fracture mechanics is computationally expensive and hence, cannot be directly applied to large components or systems crucial for many applications. Rather, the physics on these scales must be averaged or scaled-up and incorporated into continuum models used at the macroscale. Again, on the continuum scale, it becomes a computational burden to account for the complexities of fracture networks, including all possible fracture orientations and lengths, so significant sub-scale information is often lost when moving upwards in length scales.

In this manuscript we take advantage of recent advances in machine learning to develop a computationally efficient, reduced order model of a high fidelity model that accounts for the evolution of discrete fractures. The high fidelity model includes the majority of the physics necessary to predict failure, but is computationally prohibitive and too data intensive to be effective at the macroscale (e.g. several meters and greater). We demonstrate that our reduced order model mimics the high fidelity model for a given set of initial and boundary conditions at 2-4 orders of magnitude speed up. The eventual goal is to use the workflow demonstrated in this manuscript to incorporate sub-scale information using our efficient machine learning emulators into macroscale continuum constitutive models for more accurate failure predictions.

## 2. Background

A common approach for macroscale continuum failure models is to utilize average mesoscale quantities informed from higher fidelity modeling approaches where discrete fracture evolution can be captured, including mechanisms such as, velocities of tip growth, process zone stress states, and/or coalescence. The lower length scale models are used to derive upscaled relationships and determine quantities such as effective moduli to be used in the continuum scale model [14, 15, 16, 17].

Analytical models for fracture propagation have also been valuable for modeling and understanding fracture growth since they are computationally inexpensive and are connected to physical insight. In general, analytical approaches, including that of Griffith [9, 10] and Paris [11], must make broad assumptions in order to remain tractable. Hence, they neglect complexities such as fractures coalescence, variable orientation, and fracture nucleation and focus on understanding how pre-existing fractures grow due to the stress state present, and contain no interaction terms with other fractures. These assumptions can have a large impact on results since many materials, without a dominant fracture in the macroscale, have been shown experimentally and numerically to interact extensively before failure is reached. High fidelity numerical models provide an alternative to these analytical models allowing for the majority of physics to be considered but these models are often computationally prohibitive at the macroscale. Currently a popular approach for modeling fracture dynamics is with the use of Finite Element Methods (FEM) [18] or Finite Discrete Element

Methods (FDEM) [19]. Advantageously, both methods can account for individual fractures, and interactions between many fractures within a network. The key differences between FEM and FDEM methodologies is in how the cracks propagate, along edges with voids representing cracks (FDEM) or moving the crack boundaries through the mesh using cohesive elements, mesh refinement, or other numerical methods (FEM). The obstacle to properly modeling failure aspects at the macroscale such as time to failure or damage evolution lies in finding the connection between how lower scale phenomena affect the material strength and damage. The geometry, orientation, and size of individual fractures are a few of the many mesoscale features that influence macro-scale behavior.

The high fidelity model we have chosen to model fracture propagation to inform the ML approach is the Hybrid Optimization Software Suite (HOSS), a FDEM analysis tool that can account for the complexity of a fracture network's growth over periods of time [20]. This approach can result in billions of unknowns for a relatively small system ($10^6$ fractures) resulting in a computationally intractable problem at larger scales of interest to many applications. For example, running a 3D simulation of geomaterial on the scale of $1m \times 1m \times 1m$ with a fine enough mesh to resolve individual fractures would require about $500,000$ CPU hours. Thus, bridging the gap between these two scales is a major obstacle.

In order to circumvent the cost of running high-fidelity models like HOSS directly at the macroscale to model a large spatial and temporal region, we have developed a machine learning (ML) approach to learn and abstract the information that HOSS can provide, but at a much lower computational cost. Our long-term goal is to use ML as the bridge that spans the gap between meso- and macroscales for failure prediction. ML is a field that deals with the design, development and implementation of techniques that permit computers to learn based on data [21], [22]. There have been successful applications of ML across many fields like natural language processing [23], object recognition [24], and bioinformatics [25]. ML is classified into three categories based on the objective of the technique; supervised ML, unsupervised ML and reinforcement learning. Of these techniques, supervised ML has been employed because of its focus on finding a mathematical function that maps inputs to outputs. Examples of supervised ML techniques include random forests (RF) [26], support vector machines (SVM)[27] and artificial neural networks (ANNs) [28].

In the context of materials, ML has recently been used [29] to tune mesh-based parameters in an FEM model such as element stiffness, number of elements, etc. to match experimental data of fracturing in a small steel frame. Using neural networks, they achieved a 10% increase in accuracy in simulation output compared to experimental results, but at the cost of substantial growth in computational burden. Additionally for this approach to be successful, a large amount of experimental data is required to ensure over-fitting of the neural network doesn't occur. Over-fitting is when the ML model begins predicting noise or errors in the data, usually due to the high number of trainable parameters compared to observations available. In contrast to the above approach, we have generated an ML method that is trained to learn the physics of fracture propagation and interaction leading to coalescence and eventually, failure based on hundreds of HOSS simulation outputs. We extract vital patterns and trends from the high-fidelity data to build a predictive model that can emulate the HOSS simulations in a fraction of the time.

The remainder of this article is organized as follows. We present a brief description of the FDEM software HOSS in Section 3, and outline our ML methodologies in Section 4. Section 5 contains the Results and Discussions of the different ML methods compared against HOSS and discuss our conclusions in Section 6.

## 3. FDEM model: HOSS

HOSS (Hybrid Optimization Software Suite) is a multi-physics numerical tool based on the combined finite-discrete element method (FDEM). The FDEM was proposed in the early 1990s as an alternative to describe the transition from continuum to discrete material behavior that occurs upon failure, i.e., fracture and fragmentation processes of brittle geologic materials. One of the key aspects of FDEM formulation is the minimization of the assumptions made regarding the behavior of the material; fracture and fragmentation processes are described explicitly based on conservation laws. In an FDEM framework the solid domains (called discrete elements) are discretized into finite elements. Cracks must be finely resolved spatially, with dozens to hundreds of finite elements along the length of each crack. The governing equation of the FDEM systems are based on Newton's laws [30] and are solved by using an explicit (time marching) central difference method, which makes it necessary to use very small timesteps to update the system state dynamically. As a result, even simulations involving laboratory size samples with thousands of microcracks can result in petabytes of data. Additionally, since the initial state of defects and microcracks in the samples can never be known exactly, predictive capability in an uncertainty quantification framework requiring thousands of simulations is computationally intractable. A full description of the method is outside the scope of this paper; however, the interested reader can refer to the following comprehensive references for more details: [30, 31, 32].

## 4. Methods

In this section, we describe the ML approach to extracting, learning and predicting material behavior. The data is extracted from a set of 200, 2 dimensional HOSS simulations of a geomaterial that is 2 meters by 3 meters. These geomaterials contain 20 randomly distributed fractures that have a uniform length of 30 centimeters. The lengths of initial fractures were constricted to uniform lengths since larger lengths of initial fractures tend to dominate fracture propagation and failure. Our interest was in fracture coalescence, thus eliminating the dominance of longer fractures with uniform lengths gave us a clearer view of this phenomena. These initial fractures varied between three orientations, 0, 60 and 120 degrees to the applied load on the material. The material is pulled from the top at a constant velocity of 0.1 meters per second, the bottom boundary is fixed. The simulation ended when a continuous fracture path connected two opposite boundaries of the material (i.e., when the material split into two separate pieces) – we call this "material failure". At the point of failure, the material is unable to bear any further load.

A major challenge with applying ML to predict fracture growth in materials is engineering features that can generalize to a wide array of fracture scenarios. We isolate pairs of neighboring fractures and utilize their properties and interactions as training data for the ML model. Extracting neighboring pairs of fractures incorporates the local effects of fractures, but enables us to make some predictions on the global material. In addition, fracture pair information can be stored and used on any future simulation regardless of the size of material, number of fractures or strain rate (these variables are present in the extraction and learning of pairwise fracture interactions). Every fracture pair will have a feature vector of fracture information (i.e. lengths, orientations and distance between the pair of fractures) that is believed to be vital to coalescence. This ML method focuses on pairwise interactions of initial fractures and their corresponding geometric features. Therefore, the trained model on this data could be employed to predict fracture coalescence on similar systems (domain size, loading conditions, etc.) with a variable amount of fractures. With that being said, compressive loading and domain sizes that vary widely from the above initial conditions would require datasets that are more closely related to those conditions. We have taken the representative feature vectors of all the fractures in a given simulation and then grouped them by pairs that are given labels indicating whether they coalesced or not over the simulation's runtime. In the jargon of supervised ML, the feature vector is the input to the ML model and the ML model maps the feature vector to a label. The ML model is then trained on a set of feature vectors and corresponding labels (we call the combination of a feature vector and label a data point) so that this map is consistent with the data. Typically, the data is split into two sets where one is used to train the ML model and the other is used to test the ML model. RFs, DTs and ANNs were the chosen ML algorithms because of their relative ease to train and predict with. These algorithms have been given a relatively small depth to see how well they can learn patterns in the data without over-fitting from an abundance of trainable parameters.

From each of the possible fracture pairs, we extract several features including the length of the two fractures (denoted $L_1$ and $L_2$), the orientations (denoted $\theta_1$ and $\theta_2$), the distance between the two fractures (denoted $Dist$), and the minimum distance from one of the fractures to the nearest boundary (denoted $min\_DB$). These fracture pair features would then be given a label indicating whether or not they coalesce We also create a second data set for the fracture pairs that coalesce, where the labels here are the amount of time that elapsed before they coalesced. Each simulation has 20 initial fractures which yield 190 unique candidate pairs of fractures ($\frac{N!}{(N-2)!\cdot 2!}$), resulting in a total of 5,200 data points for the entire dataset. This data implicitly contains a wealth of information related to fracture interactions, fracture propagation speeds, and the influence of orientations on fracture dynamics, but is much more concise than the detailed information needed to perform the full HOSS simulation (e.g., high-resolution stress fields, nodal velocities, etc.). From each HOSS simulation, we also extract the time of material failure. While the time of material failure is not used for training purposes, we use our ML models to predict this and it serves as an independent validation of our ML approach.

Our ML models are trained to predict whether or not two fractures coalesce, and how long it takes for them to coalesce. The other features ($L_1$, $L_2$, $\theta_1$, $\theta_2$, $Dist$, and $min\_DB$) are used to make these predictions (see Figure 1). Using these ML models, we are also able to make predictions about the materials path to failure (i.e., a sequence of fractures that coalesce amongst each other as well as the left and right boundaries), and the time that elapses before the material fails.

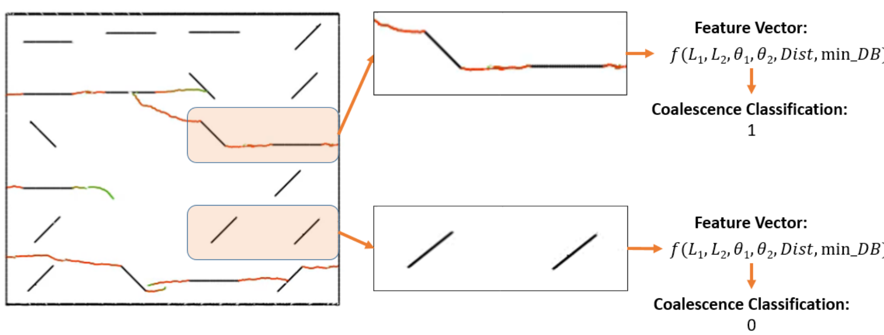


Figure 1: Visualization of the extraction of pair data from a 2D HOSS fracture simulation. In the feature vector, the $L$s are lengths of initial fractures, the $\theta$s are fracture orientations, Dist is the distance between the two fractures and min_DB is the minimum distance to the boundary from either fracture. The simulation above has 20 initial fractures which yield 190 data points (with boundary-fracture pairs included).

RFs, DTs and ANNs are popular supervised ML methods that have been used successfully on a vast range of datasets[33, 34]. RFs and DTs work by hierarchically branching based on the feature vector to minimize a cost function. The method is most effective when after branching through the tree, the data becomes homogeneous or nearly homogeneous. The branching is represented by a tree-like graphical model. When making a prediction with a DT, the algorithm starts at the top of the tree and progressively moves down the tree until reaching a leaf node. Non-leaf nodes in the tree contain conditional statements where if the condition is satisfied, the algorithm follows one branch of the tree and follows another branch otherwise. The leaf nodes contain a prediction. In classification problems, these predictions will be a class (e.g., a binary value indicating whether or not two fractures coalesce). In regression, the predictions will be some continuous quantity of interest (e.g., the amount of time it takes for two fractures to coalesce). Figure 2 illustrates a DT that estimates the time of material failure for a simulation containing 3 fractures using ratio of fracture lengths and vertical distance between fractures as features. In some cases, a probabilistic approach is used where the leaves contain probabilities of predictions rather than predictions themselves.

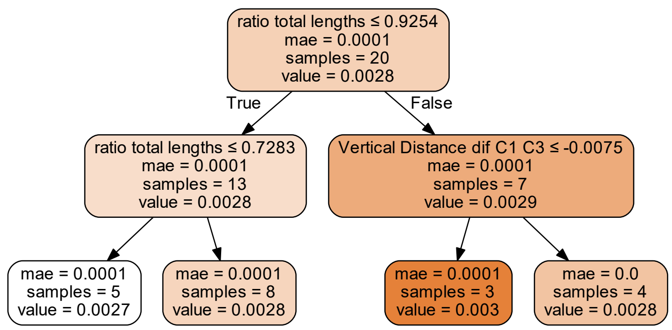


Figure 2: A simple schematic of a trained DT with a depth of 2 predicting the time of failure for 20 different simulations or samples. The features in this problem are the ratio of the total lengths of two fractures and the vertical skew or distance between the tips of the two fractures.

RFs are an ensemble learning method which means that they use a collection of ML algorithms to increase accuracy. RFs employ a multitude of DTs that are each trained on different subsets of the training dataset. The prediction from each of the DTs are collectively used to make the prediction for the RF (usually using the mean or majority vote). These ensemble learning approaches have been shown to substantially increase accuracy compared to a lone ML algorithm[33].

ANNs train and predict in a completely different fashion compared to RFs and DTs. ANNs are composed of layers of neurons. These neurons are connected to the nearest previous and subsequent layer of neurons. These connections have an associated weight that is adapted and learned during the training of the network. Each neuron has an activation function (functions can vary based on user input) that will output a value based on the weighted input. The network's trainable parameters (weights, bias in the activations function, etc.) are updated using the back propagation algorithm which is governed by the error between predicted values from the network and actual labels[35].

Development of the architecture of an ANN is a highly researched and important aspect of deep learning. One concern of a large network is that if it has more parameters than training datapoints it's highly likely to be subject to over fitting. The particular structure of our ANN was relatively small with four layers of 12, 8, 4 and 1 being the number of neurons in each layer. Decreasing the number of neurons from one layer to the next is common practice in ANN creation. This structure siphons number of inputs gradually without large amounts of data loss at any particular stage.

All the ML algorithms were first trained on all the data points and classified to indicate whether a fracture coalesced or not (classification models). Subsequently, all data points that had coalesced were fed into similarly constructed ML algorithms and trained to predict when these fractures coalesced (regression models). Thus, feeding the trained classification and regression networks a set of initial fracture pair features would output predictions of where and when fractures coalesce. The time of material failure can then be determined by computing the earliest time at which a set of fractures containing connections to both the left and right boundary have coalesced. Our workflow from training data through coalescence predictions and material failure is shown in figure 3.

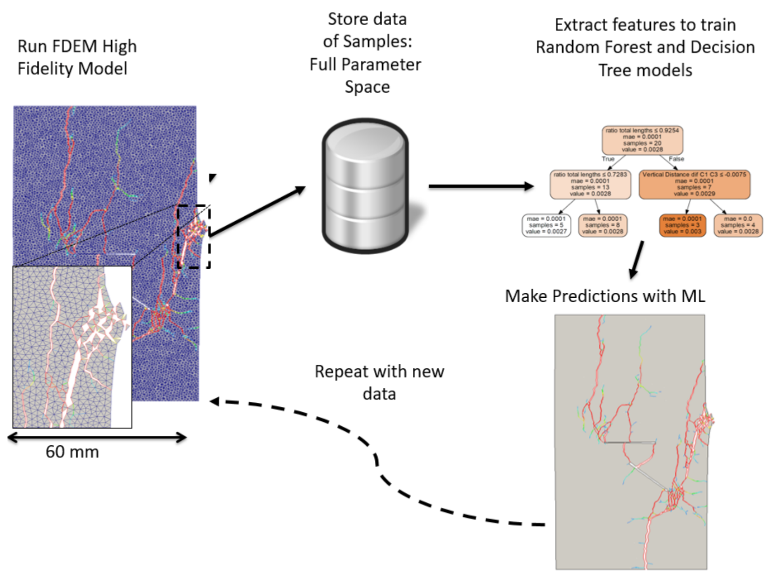


Figure 3: Our illustrated workflow from a FDEM mesh to the storage of the corresponding data. Feature selection and model training is then done on the stored data and prediction are made on held out simulation data. Once this framework is built, additional data can be used for training with varying initial HOSS conditions.

## 5. Results & Discussion

Results from the neighboring fracture approach are judged by two criteria, the location of the dominant fracture path and the time of material failure. The classification network, which determines if fractures unite based on their features, was trained first. The data was split into training and testing sets with 80% of the data be used for training at 20% of the data being used for testing. Random shuffling of the data and the mean percent error (MPE) cost function were applied. Given a feature vector from a pair of fractures the model will predict if those two fractures will coalesce or not. Table 1 shows the probability of misclassifying a pair of fractures as coalescing or not.

| | Mean Percent Error: Decision Tree | Mean Percent Error: Random Forests | Mean Percent Error: Neural Network |
|---|---|---|---|
| Testing set 1 | 3.33% | 2.78% | 2.11% |
| Testing set 2 | 3.25% | 2.32% | 1.98% |
| Testing set 3 | 3.63% | 2.15% | 2.17% |
| Testing set 4 | 3.27% | 2.03% | 1.92% |
| Testing set 5 | 3.57% | 2.88% | 2.63% |

Table 1: The percent error of predicting fracture coalescence on a particular test set. The training data sets were split into 5 equivalently sized, randomly distributed data points This ensures generality of the ML model. DT, RF and ANN had correlation coefficients of 0.79, 0.85 and 0.86 respectively.

For fractures that have been determined to coalesce, the next step is to understand how long it takes for them to coalesce. This model is trained on all the pairs that coalesced from the classification model. The depth for the DT and RF models were set at 7, which corresponds to 255 possible splits or attributes. The feature vectors are not changed between the coalescence classifier and the time-of-coalescence regression – only the labels have changed. The labels for time-of-coalescence model are the times at which the pairs coalesced. The training split is maintained at 80% and the depth is decreased to 5 (63 possible splits or attributes), since the total number of data points has decreased. The RFs have 10 individual estimators (decision trees) that are trained and used for testing. Since the output is no longer binary, the cost function has been changed to Mean Absolute Error (MAE). The standard deviation of the time of coalescence was 0.00121 seconds and the average time of coalescence was 0.00250 seconds. The MAE for the RF model was around 0.000412 seconds, or about 16.5% of the average time of coalescence

This ML approach was tested on fracture configurations that were held out of the training data. A typical result is shown below (Figure 4) between the HOSS and ML model at two different times. The ML model has shown that it can capture the failure dynamics well enough to predict the failure path of the material, but missing some fracture connectivity in other regions of the domain.

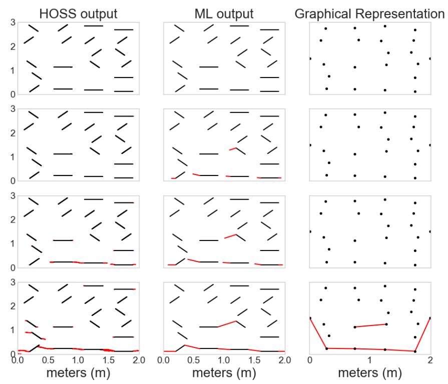


Figure 4: The left most plots are of the HOSS simulation at an intermediate time step (top left) and at failure (bottom left). The middle figure is the ML predicted results at the two times. The far right plots are the graphical representations of both models.

Our ML models have been tested against one another and the ground truth, HOSS. The three crack growth approaches were tested on 10 randomly chosen fracture configurations to find the time of material failure (results shown in Figure 5). An additional post-processing step was required in cases where the ML model did not predict a path to failure. In these cases, all fractures that have began propagating are fractured further until a failure path has occurred. This is done by using the average velocity of the fracture growth up to this point in the simulation. These averaged fracture velocities are then applied to the existing fractures within the material and growth follows the shortest path to the boundary.

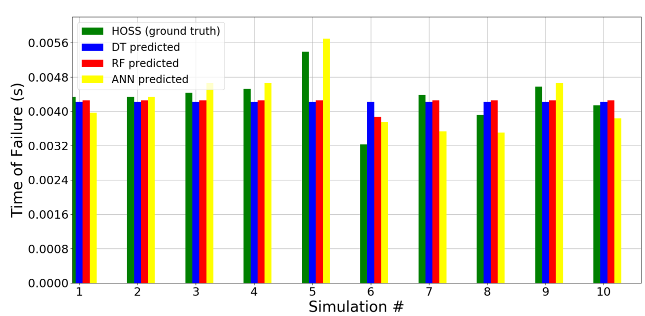


Figure 5: Predicted times to failure for 10 randomly chosen simulations of the ANN (yellow), Decision Tree of depth 5 (blue), Random Forest of depth 5 (red) and the ground truth, HOSS (green).

The correlation coefficients between ANN, RF and DT with the HOSS time of failure are 0.68, 0.68 and 0.42, respectively. In addition, DTs and RFs of depth 5 have only about 63 tunable parameters for training so over fitting is unlikely on a training dataset with over 20,000 data points. We recognize that the correlation and accuracy was greatly reduced between the spatial aspects of failure and the temporal. We believe this is due to ignoring information like stress and nodal velocities. These values are precursors to fracturing and without utilizing this data in our model, the growth and coalescence of fractures was unable to be learned with great accuracy. Since the model has reasonably good correlation on the testing set (which was not used for training) and the number of tunable parameters is far less than the number of data points, we conclude that generalization is present. In the context of ML, generalization (which is one of the most important qualities of a successful ML model) is the ability for the model to make accurate predictions on data that was not used for training.

An interesting and problematic outcome of including all the possible pairs of fractures as data points is imbalance between the frequency of coalescence and non-coalescence. Figure 2 illustrates a typical simulation where only 5-9 of the possible 190 fracture pairs in a simulation coalesce. That is, the number of pairs which do not coalesce greatly exceeds the number of pairs that do coalesce. From the 50 simulations used for testing, there was a total of 9,500 data points (pairs of fractures), where 139 of these pairs coalesced and 9,361 did not coalesce. Thus, about 1.5% of the data points are labeled as coalescing and around 98.5% of the data points are labeled as non-coalescing. This uneven spread of labels is a problem in the ML field called class imbalance. In these situations, ML algorithms tend to get easily caught in the local minimum of predicting the heavily weighted class (non-coalescing labels in this case). If the ML model predicted that no fracture pairs would coalesce this would yield a very low error (1.5% in this case) because of the imbalance.

Many methods have been created to combat the problem of class imbalance [36]. One method to eliminate this imbalance is by reducing the number of overly represented labels from the dataset. This method solves the imbalance problem but also could get rid of a substantial amount of important data for future training and testing. Another approach is to weight the underrepresented labels higher when training (i.e., apply a greater penalty for failing to correctly predict the underrepresented label). This solves the class imbalance problem while incorporating all the available data. When employing this tool, an addition of complexity in training models comes about from iteratively testing various weights. This is illustrated in Figure 7 where five different weights are used in the training of identical ANNs. These graphs are called confusion matrices and visualize the number and type of predictive errors that occur during testing.

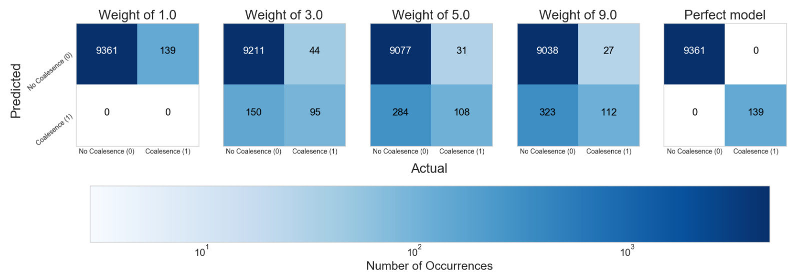


Figure 6: Confusion matrices of identical ANNs trained with an increasing weighting on coalescence labels (from left to right). In each matrix, the top left represents the number of true negative labels, top right are false positive labels, bottom left are false negative labels and the bottom right are true positive labels. A perfect model would have the bottom left to top right diagonal have zero values within them. This is illustrated in the farthest right confusion matrix.

As can be seen in Figure 6, when the weight is increased, the number of cases in which the model correctly predicts coalescence increases. A side effect of this weighting scheme is the decrease in accuracy of non-coalescence predictions. There is a trade-off in performance of these two regimes, but choosing an appropriate weight yields a significant improvement in predictive power.

The computational cost for the ML models are negligible (under a tenth of a second to compute on a 20 fracture system and requires less than a megabyte of storage) in comparison the HOSS model. A single HOSS run for the models used here takes about 4 hours on 400 cores. Without taking into consideration the 400 processors necessary for a HOSS run, this is still over a 5 magnitude speed up in time.

## 6. Conclusions

ML provides a new path forward for accurately resolving some of the dynamics of fracture propagation in brittle materials. In this article we introduced a simple learning approach to predict the location and timing of fracture coalescence for a brittle material. This approach was a proof of concept for ML on fracture dynamics and we ultimately plan to both increase the size and improve the accuracy of the model to enable us to accommodate more diverse materials, under more diverse conditions, and with more fractures. FEM models like HOSS have proven their worth in the domain of dynamic material simulations, but scaling to larger, more complex problems is often computationally prohibitive due to the difference in length scales between small fractures and bulk material sizes.

RFs and DTs were successfully employed to extract key patterns and trends in the exploration of the brittle material failure dataset, while providing an accurate predictive model. This technique utilized only 63 trainable parameters, yet predict the time to failure of various material fracture distributions within 5% of the ground truth model. In addition, a 4 orders of magnitude reduction in the storage of data and 5 orders of magnitude drop in computational time has been achieved (compared to the HOSS model). Discovering the set of features that led to the current accuracy of the ML model will aid in engineering features for larger and more complex ML algorithms.

As stated previously, our algorithm only considers pairwise interactions, neglecting the effects of 3 and higher order crack interactions. The assumption holds good for the initial and boundary conditions considered in this example, as illustrated on Fig 5. One possible explanation as to why the pairwise terms were sufficient to model crack propagation is the low density of initial cracks. Additionally, the low strain rate loading condition resulting in Mode I failure favors crack propagation in the horizontal direction, perpendicular to the loading. This would favor pairwise interactions between cracks laid out along the same row over higher order terms. As part of our future work, we will include 3 and 4 crack interaction terms to test our hypothesis. We also plan to broaden the scope of our algorithm by increasing the loading rates and crack densities.

Current microscale models such as HOSS are too computationally demanding to be coupled with macroscale models that cannot account for individual fractures. This work is a stepping stone toward the development of accurate microscale models that are computationally efficient enough to be coupled with macroscale models. More work is needed to realize this end goal, but we have demonstrated that ML holds great potential for filling this role.

## Acknowledgement

The research presented in this article was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project number 20170103DR. Authors thank the LANL Institutional Computing program for their support in generating data used in this work.

## References

[1] D M Dimiduk, D B Miracle, and C H Ward. Development of intermetallic materials for aerospace systems. *Materials Science and Technology*, 8(4):367–375, 1992.

[2] D. O'Malley, S. Karra, R. P. Currier, N. Makedonska, J. D. Hyman, and H. S. Viswanathan. Where Does Water Go During Hydraulic Fracturing? *Groundwater*, 54(4):488–497, 2016.

[3] K. Ohji. Introduction to Fracture Mechanics. *Journal of the Society of Materials Science, Japan*, 32(359):935–941, 1983.

[4] E. Bouchbinder, T. Goldman, and J. Fineberg. The dynamics of rapid fracture: instabilities, nonlinearities and length scales. *Reports on Progress in Physics*, 77:1–30, 2014.

[5] G.T. Camacho and M. Ortiz. Computational modelling of impact damage in brittle materials. *International Journal of Solids and Structures*, 33(20-22):2899–2938, aug 1996.

[6] G.P. Cherepanov. *Mechanics of Brittle Fracture*. Mcgraw-Hill, 1979.

[7] S. Y. Xie and J. F. Shao. Experimental investigation and poroplastic modelling of saturated porous geomaterials. *International Journal of Plasticity*, 39:27–45, dec 2012.

[8] T GATES, G ODEGARD, S FRANKLAND, and T CLANCY. Computational materials: Multi-scale modeling and simulation of nanostructured materials. *Composites Science and Technology*, 65(15-16):2416–2434, dec 2005.

[9] G I Barenblatt. The Mathematical Theory of Equilibrium Cracks in Brittle Fracture. In H L Dryden, Th. von Kármán, G Kuerti, F H van den Dungen, and L Howarth, editors, *Advances in Applied Mechanics*, volume 7 of *Advances in Applied Mechanics*, pages 55–129. Elsevier, 1962.

[10] A. A. Griffith. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of London*, 221:163–198, 1921.

[11] P. Paris and F. Erdogan. A Critical Analysis of Crack Propagation Laws. *Journal of Basic Engineering*, 85(4), 1963.

[12] B.N. Cox, H. Gao, D. Gross, and D. Rittel. Modern topics and challenges in dynamic fracture. *Journal of the Mechanics and Physics of Solids*, 53:565–596, 2005.

[13] M. Marder. Particle methods in the study of fracture. *International Journal of Fracture*, 196:169–188, 2015.

[14] I. M. Gitman, H. Askes, and L. J. Sluys. Coupled-volume multi-scale modelling of quasi-brittle material. *European Journal of Mechanics, A/Solids*, 27(3):302–327, 2008.

[15] Adnan Ibrahimbegovic and Arnaud Delaplace. Microscale and mesoscale discrete models for dynamic fracture of structures built of brittle material. *Computers and Structures*, 81(12):1255–1265, may 2003.

[16] J. W. Ju and T. M. Chen. Effective elastic moduli of two- dimensional brittle solids with interacting microcracks, part i: Basic formulations. *Journal of Applied Mechanics*, 61:349–357, 1994.

[17] L. G. Margolin. Elastic moduli of a cracked body. *International Journal of Fracture*, 22:65–79, 1980.

[18] A. Needleman. Computational mechanics at the mesoscale. *Acta Materialia*, 48(1):105–124, jan 2000.

[19] A. Hillerborg, M. Modéer, and P. E. Petersson. Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements. *Cement and Concrete Research*, 6(6):773–781, nov 1976.

[20] Antonio A Munjiza. *The Combined Finite-Discrete Element Method*. John Wiley & Sons, London, 2004.

[21] Tshilidzi Marwala. *Finite-element-model updating using computional intelligence techniques: Applications to structural dynamics*. Springer-Verlag London, 2010.

[22] Field Cady. Machine Learning Overview. *The Data Science Handbook*, pages 87—-91, 2017.

[23] Gerhard Weikum. Foundations of statistical natural language processing. *ACM SIGMOD Record*, 31(3):37, 2002.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[25] Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, and Victor Robles. Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112, 2006.

[26] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[27] Simon Tong and Daphne Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, pages 45–66, 2001.

[28] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, jan 1989.

[29] J L Zapico, K Worden, F J Molina, Bin Xu, Zhishen Wu, Koichi Yokoyama, M A Yunus, Abdul Rani, H Ouyang, A González-Buelga, M P González, and R Alonso. Finite element model updating of a small steel frame using neural networks Damage assessment using neural networks Identification of Damaged Spot Welds in a Complicated Joined Structure Finite element model updating of a small steel frame using neural net. *Smart Mater. Struct. Smart Mater. Struct*, 17(17):45016–11, 2008.

[30] Antonio A Munjiza, D.R.J. Owen, and N. Bicanic. A combined finite-discrete element method in transient dynamics of fracturing solids. *Engineering Computations*, 12:145–174, 1995.

[31] Z. Lei, Esteban Rougier, Earl E Knight, Antonio A Munjiza, and Hari Viswanathan. A generalized anisotropic deformation formulation for geomaterials. *Computational Particle Mechanics*, 3:215–228, 2016.

[32] Esteban Rougier, Earl E Knight, S.T. Broome, A.J. Sussman, and Antonio A Munjiza. Validation of a three-dimensional finite-discrete element method using experimental results of the Split Hopkinson Pressure Bar test. *International Journal of Rock Mechanics and Mining Sciences*, 70:101–108, 2014.

[33] David Opitz, Richard Maclin, David Optiz, and Richard Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligent Research*, 11:169–198, 1999.

[34] O. Rokach, Lior; Maimon. *Data mining with decision trees: theory and applications*. World Scientific Pub Co Inc., 2008.

[35] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[36] Charles X Ling and Victor S Sheng. *Class Imbalance Problem*, page 171. Springer US, Boston, MA, 2010.