Deep Interval Neural Network in Computational Mechanics

David Betancourt[1,2)] and Rafi L. Muhanna[1,2)]

[1)] *School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA, david.betancourt@gatech.edu* [2)] *School of Civil and Environmental Engineering,*

Georgia Institute of Technology, Atlanta, GA 30332, USA, rafi.muhanna@gatech.edu

Abstract. In computational mechanics, when the input properties to the finite element (FE) model contain uncertainty, it is necessary to model the spatiotemporal variation of the uncertainty and to make predictions with an FE model that can propagate this uncertainty in the system. Often times, we do not have enough information to assume a probability distribution for the input, but only have access to the upper and lower bounds of the data. In such cases, when only the bounds of the input data are known, interval analysis (IA) offers an alternative to model the uncertainty. In addition, we generally do not have direct access to the uncertain property (e.g., elastic modulus) but only to other indirect observations that help characterize it (e.g., strain measurements). As a result, we present an interval deep learning algorithm to model the uncertainty of the input properties for the Interval Finite Element method (IFEM). For this work, we developed a deep interval-valued neural network (DINN) to quantify interval uncertainty of IFEM inputs. The DINN takes as input indirect observations with interval uncertainty (e.g., sensor measurements) and predicts the spatial and/or temporal variation of an unknown IFEM interval input. In this way, interval uncertainty *of the system inputs* is quantified by the DINN and the interval uncertainty *of the mechanical response* is quantified by the IFEM. A numerical experiment is conducted using a dataset of concrete mix measurements with interval uncertainty to predict concrete strength. We conclude by showing that our data-centric method achieves good results compared to ground truth values of the material properties.

Keywords: deep learning, interval uncertainty, interval finite element, interval deep learning

1. Introduction

The analysis and design of complex engineering systems is exposed to a great number of uncertainties in the inputs of the systems' models. Not modeling this uncertainty properly can lead to inaccurate predictions of the system's performance, which in turn can lead to disastrous outcomes. In computational solid mechanics, a system analysis is typically performed with a finite element (FE) model. The FE model receives as input the physical properties of the system under consideration (e.g., material properties, dimensions, support conditions, and applied loading). As output, the FE model computes the resulting responses of the system (e.g., displacements, stresses, and vibration frequencies). The described process is generally referred to as the *forward problem*. When the input properties to the FE model contain uncertainty, it is necessary to model the uncertainty and to make predictions with an FE model that can propagate this uncertainty in the system.

In general, the finite element method (FEM) does not have an integrated mechanism to model the spatial and/or temporal (spatiotemporal) uncertainty variation of the input properties. Given this limitation, the input uncertainty is commonly modeled by either assigning the same FE input property over the entire FE mesh or on an element-basis by the FE analyst, but without an analytical field that considers the spatiotemporal dependencies between mesh elements. Such approaches might vary from the economical to the conservative but they are not always unequivocal. Uncertainty in the input must be duly considered in order to guarantee strength, serviceability, and economics of the structural system being analyzed, as well as the aggregate environmental impact of overly-conservative structures because of excess material. In fact, there are three important considerations for uncertainty quantification in the input properties of the forward problem. First, the spatiotemporal dependence in the domain must be considered—also known as the uncertain field. Second, in many applications in solid mechanics we do not have access to directly measure an unknown property but only have access to indirect measurements. Thus, we need a way to infer the input variable from indirect observations. Third, we need a way to model the aleatory and epistemic uncertainty in the input properties.

Some probabilistic paradigms, such as the Stochastic Finite Element Method (SFEM) (Ghanem and Spanos, 2003), integrate Gaussian random fields with the deterministic FE method to model the uncertain field (i.e., the spatiotemporal-varying uncertainty) in the system inputs and propagate the input uncertainty through the FE model. Nonetheless, random field theory is inadequate for cases where there is epistemic uncertainty and/or not enough data to select a probability distribution (Zhang, 2005). Under such cases, the Interval Finite Element Method (IFEM) presents an alternative paradigm to model system uncertainty because it does not require assumptions on a probability model. Muhanna and Mullen (Muhanna and Mullen, 2001) have shown that the IFEM can be duly beneficial and efficient in computational solid mechanics because there is usually not enough knowledge about the probabilities of the inputs.

However, as with the deterministic FEM in general, the IFEM does not have an integrated procedure to model the uncertainty in the system inputs. As a result, *interval field* models have been recently proposed (Moens and Hanss, 2011; Sofi, 2017; Wu and Gao, 2017) in order to model the uncertainty field for the IFEM input. Previously, we presented a real-valued deep neural network method to model the spatiotemporal variation of an uncertain input IFEM property as a function of indirect observations ((Betancourt et al., 2018)). The resulting output of the real-valued deep neural network could then be given postulated interval bounds based on expert knowledge. For this work, we developed a novel interval-valued deep neural network—Deep Interval Neural Network (DINN)—capable to quantify the aleatory and epistemic uncertainty in the input by using IA. The DINN is able to predict an interval field by using interval-valued indirect measurements of an uncertain IFEM input property.

Our contribution with the DINN is to address the three concerns for uncertainty quantification of the input properties of the FE model: (i) it quantifies the uncertain field by means of an interval field; (ii) it infers the input property (e.g., material properties) by indirect observations (i.e., sensor measurements); and (iii) it quantifies the aleatory and epistemic uncertainty in the input by using IA. Additionally, the DINN can quantify the parameter uncertainty of the neural network itself as part of the algorithm.

This paper is organized as follows. On Sec. 1.1 we explore related work in terms of interval field, interval neural networks, and general data-driven methods in computational mechanics. Sec. 2 explains the process of input uncertainty quantification in the structural forward problem and introduces key concepts of IA. Sec. 3 explains in detail the development of the DINN. Sec. 4 introduces the concept of the Supervised Interval Field (SIF). Finally, Sec. 5 illustrates the overall process through a numerical experiment conducted using a dataset of concrete mix measurements with interval uncertainty to predict concrete strength.

1.1. Interval Field Related Work

The first to incorporate the concept of spatially varying input uncertainty within the IFEM was in (Moens et al., 2011), with subsequent method updates in (Verhaeghe et al., 2011; Imholz et al., 2016). In (Imholz et al., 2016) an interval field concept was developed which allowed to use spatial dependency between elements, which is otherwise not possible using interval parameters. To that end, two methods were introduced: the Local Interval Field Decomposition (LIFD), and the Inverse Distance Weighting Interpolation (IDW). For the LIFD, the dependency parameters are determined from a gradient method developed by Imholz et al., which defines the dependency between parameters as the maximum difference that can occur between them. In this definition, perfect dependency corresponds to a zero gradient and perfect independency corresponds to the maximum gradient between parameters. The dependency parameters are discretely defined at each element of the mesh—thus the dimensionality of the interval field equals the numbers of elements in the mesh. The discrete values for the dependency parameters are determined by a smooth second-order polynomial function presented in (Imholz et al., 2016). For the IDW method, the base functions needed for the interval field are determined and spatially selected based on prior engineering knowledge.

The LIFD is computationally expensive as the dimensionality of the uncertainty equals the number of finite elements in the mesh. The IDW is computationally cheaper, and could produce good results where enough a-priori domain engineering knowledge is available. As a result, expert hand-engineering would be required to produce good results. Both methods were only presented for 1D cases.

(Wu and Gao, 2017) developed a hybrid method for static FE analysis (X-UISS method) that combines random fields and intervals fields. It has an algorithm that easily extends to 2D and 3D domains. For the interval field implementation, the X-UISS method defines the upper-bound UB and lower-bound LB functions from the extrema of a set of measurements at any spatial coordinate. Then, the UB and LB of measured points are linearly interpolated between the $\mathbf{x}$ points where data is unavailable. Then, the conventional IFEM (Muhanna and Mullen, 2001) procedure is implemented using the linearly interpolated values for the domain. The pitfall of this method is that it is computationally expensive by having to solve both SFEM and IFEM methods but more importantly because it uses linear interpolation between data points, which is an oversimplification of the spatial uncertainty variability in a domain.

(Sofi, 2017), (Sofi, 2015) developed a 1D interval field approach for IFEM which uses the extra unitary interval (EUI) in order to reduce the overestimation of the IFEM due to mutual dependency. In contrast to the Hybrid method by Wu et al. and the LIFD method by Moens et al., the dimension

of uncertainty does not depend of the FE mesh size of the model. In that work, the underlying interval field method is based on a closed-form solution of a determinate beam or an approximate solution for statically indeterminate beams. A general EUI method for 2D and 3D domains under general boundary conditions has not been developed to the authors' knowledge.

1.2. Interval Neural Networks Related Work

For deep neural networks, predictive model uncertainty quantification has customarily taken place by using Bayesian Neural Networks (BNNs) or Bayesian approximations. In (Hernández-Lobato and Adams, 2015), the authors present Probabilistic Backpropagation, a Bayesian neural network method to train deep neural networks and obtain parameter uncertainty. Shortly after, (Gal and Ghahramani, 2016) developed a method called Monte Carlo Dropout, which averages Monte Carlo simulations of predictions using Dropout (Srivastava et al., 2014) at test time in order to obtain a predictive uncertainty estimate. In deep reinforcement learning, state uncertainty in Partially Observable Markov Decision Processes (POMDPs) was studied by (Hausknecht and Stone, 2015) using Long Short-Term Memory (LSTM) units in order to *remember* past states.

On the other hand, training deep models end-to-end with uncertain input data has been less studied. Most of the work adds noise to the data with known probability distributions (Czarnecki and Podolak, 2013) or uses sampling methods (McDermott and Wikle, 2019). In many safety-critical applications, we do not have enough information to select a probability distribution for the input, as in conditions of epistemic uncertainty, but can gain access to bounds on the data by expert knowledge or by the simple nature of the data acquisition (Ferson et al., 2007). In such case, when only the bounds of the input data are known, IA (Moore et al., 2009) offers an alternative to model the uncertainty.

More generally, IA algorithms are typically used for rigorous error bounds, result verification, sensitivity analysis, and uncertainty quantification (Moore et al., 2009), (Neumaier, 1990). For uncertainty quantification in particular, IA has been used for inference with imprecise probabilities (Walley, 1991), matrix inequalities (Ben-Tal and Nemirovski, 2002), partial differential equations with finite element methods (Muhanna and Mullen, 2001), and data processing (Kreinovich et al., 2013). Recently, IA has been used as a surrogate to verify DNNs but not in a way that can be directly used for inference (Liu et al., 2019; Adam et al., 2016). Symbolic IA has also been used for security analysis of DNNs (Wang et al., 2018), but not with mathematically rigorous IA algorithms as it's used for other computational applications previously mentioned.

For neural networks (NNs) using IA for inference, prior work that began in the 1990s used interval-valued inputs for shallow NNs for smaller datasets, and with few exceptions not trained using gradient-based optimization. Nonetheless, the developments were valuable to demonstrate the concept of learning with interval-valued inputs. For example, (Ishibuchi and Tanaka, 1991; Hernandez et al., 1993) developed Backpropagation for interval arithmetic using feed-forward two-layer neural networks, (Freitag et al., 2011) developed an algorithms to train recurrent neural networks, (Freitag et al., 2012) trained a recurrent neural network using particle swarm optimization. Other work, has attempted to train neural networks with interval values but without using IA, which makes it is less rigorous (Yang et al., 2019)

In fact, some of this work carried on until recently, using older neural network architectures and optimization methods (Yang and Wu, 2012),(Yang and Liu, 2018),(Yang et al., 2019),(Huang et al., 2020). Consequently, we introduce a deep interval neural network (DINN), which is able to predict with interval-valued data and produce interval estimates using interval gradient-based optimization for deep neural networks.

1.3. Data-Driven Methods in Computational Mechanics

Recently, data-driven methods using machine learning (ML) in computational mechanics have emerged for applications from constitute modeling to damage detection. However, a deeper look at most surveyed papers revealed that the data come from simulations instead of real measured data (Korzeniowski et al., 2019), (He and Chen, 2020), (Nuttall, 2018), (Hauseux et al., 2018), with few exceptions (Reynders and De Roeck, 2014; Yeh, 2006). This is more than likely due to the high costs of obtaining data and the limited availability of public datasets in these applications. Using solely simulation data as training data for ML models is very limiting because it impairs generalization of the learning algorithm—the main goal in ML—and biases the predictions to the simulation assumptions. Nonetheless, simulation data can be in fact used to complement real measured data. Overall, it is imperative to address the challenge of gathering data for computational mechanics applications.

2. Uncertainty Modeling in Computational Mechanics

In this section, uncertainty modeling in computational mechanics is described in mathematical form as a prelude to the DINN development. In the forward problem, a system analysis is typically performed with a finite element (FE) model to determine the state of the system. The FE model receives as input the physical properties of the system. As output, the FE model computes the resulting responses of the system (e.g., displacements, stresses, and vibration frequencies).

Let's now assume that the FE inputs contain interval uncertainty and the system analysis is performed using the IFEM. The stages in the structural forward problem under interval uncertainty is shown on Fig. 1. The input uncertainty is in the spatiotemporal variation of the IFEM input variable as well as the uncertainty in the measurements. Let's also assume that we cannot directly measure the unknown input property but only have access to indirect measurements. We will quantify the input uncertainty in the described forward problem in the following fashion. Let Y be an uncertain system input to the IFEM, such as the material elastic modulus, that cannot be directly measured. Let X be a set of observations that characterize Y indirectly, such as physical properties captured by sensors. Often times, the relationship between X and Y is unknown. Other times, it is obtained through empirical engineering formulas that generally make simplifying assumptions. On the other hand, we can use a predictive model f to find the relationship between X and Y, such that $f : X \rightarrow Y$, to obtain an estimate of the sought-after property $\widehat{Y}$. The measured data for X has inherent aleatory and epistemic uncertainty due to error in the measurement devices, randomness, lack of knowledge about the data generating process, imprecision, ignorance, and poor understanding of physics phenomena. Additionally, there has to be spatial and time coherence in

neighboring points that we seek to also capture with the predictive model f. In this work, the DINN is the choice of predictive model f to quantify the uncertain input properties for the IFEM.

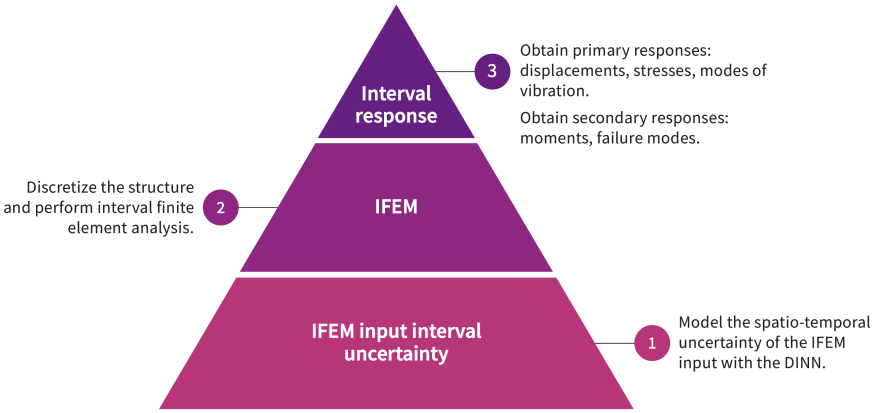

Figure 1. Stages of Forward Problem. The DINN quantifies the IFEM input uncertainty.

In solid mechanics, we often do not have enough knowledge about the probability distributions of X and Y. Indeed, an exclusively probabilistic approach would require to make assumptions on the distributions without enough knowledge, which does not always result in reliable predictions. Therefore, we instead use interval uncertainty on the values of X to obtain interval predictions Y using the DINN—which does not require probabilistic assumptions but only upper and lower bounds of the input.

3. DINN for Computational Mechanics

In this section, the details of the DINN development are explained. The goal of the DINN for this work is to predict an interval field of an IFEM input property using indirect observations $\mathbf{X}$ (i.e., indirect sensor measurements). The DINN is the interval extension of a real-valued DNN to process interval-valued matrices of any dimensionality (i.e., interval tensors). The DINN is a supervised learning algorithm in a regression setting which seeks to learn an interval predictive model $F : [\underline{X}, \overline{X}] \to [\underline{Y}, \overline{Y}] \in \mathbb{IR}$. In order to achieve this goal, the DINN is trained using the $\mathbf{X}$ interval-valued features in a d-dimensional space with n samples, along with its known $\mathbf{Y}$ interval-valued targets. This composes the training set $\mathcal{T} = \{(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)\}$ where $\mathbf{X} \in \mathbb{IR}^{n \times d}$ is the feature space and $\mathbf{Y} \in \mathbb{IR}^n$ is the target space. As its output, the DINN computes for each sample i an estimate of the interval target $\widehat{F}(\mathbf{X}_i) = \widehat{\mathbf{Y}}$. In order to do so, the training algorithm iteratively reduces the difference between the true known target $\mathbf{Y}_i$ and the prediction target $\widehat{F}(\mathbf{X}_i)$ at each sample i, by minimizing a loss function $\mathcal{L}(\widehat{F}(\mathbf{X}_i), \mathbf{Y}_i; \mathbf{W})$, such as mean squared error, parameterized by $\mathbf{W}$. Hence, we train the DINN to find the set of interval parameters $\mathbf{W}$

that minimize the loss function. After training, each new future sample i is predicted as

$$\widehat{F}(\mathbf{X}_i) = \mathbf{X}_i^{new}\mathbf{W}, \tag{1}$$

where $\mathbf{W}$ are the trained interval weights of the DINN and $\mathbf{X}_i^{new}$ is a future query of unseen values. The ultimate goal of the DINN is to generalize to predictions of unseen uncertain data where target values $\mathbf{Y}$ are not available. The following sections delve into the details of the DINN.

3.1. Fully-Connected Interval Network

For deep neural networks, the predictive interval function $\widehat{F}(\mathbf{X})$ at the output layer L_{out} is composed of the functions of the hidden layers, such that

$$\widehat{F}(\mathbf{X}) = F^{(L)}\Big(F^{(L-1)}\big(F^{(L-2)}\ldots\big(F^{(\ell)}\ldots\big(\widehat{F}(\mathbf{X})\big)\big)\big)\Big), \tag{2}$$

for $\ell = 1\ldots L$, where ℓ is the layer number.

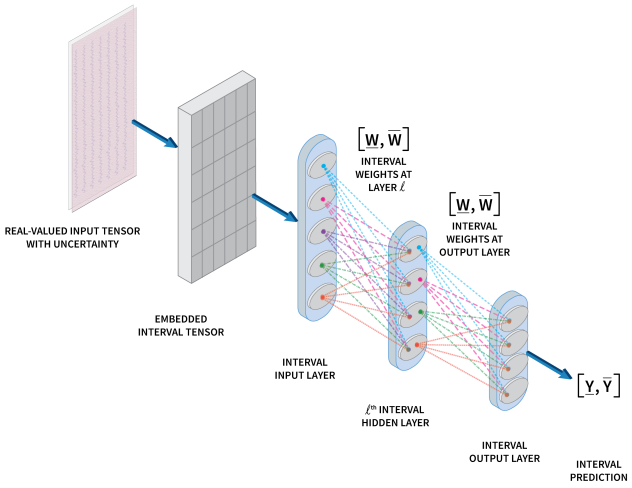

Figure 2. Schematic Representation of DINN.
Nodes represent the activations from previous layers after passing through a `ReLU`
The edges represent the interval weights of the layers.

Hence, the DINN is a composition of interval-valued functions where each of these functions is the output of each of the l layers of the network after passing through a nonlinear activation function. A schematic representation of the DINN is shown on Fig. 2. Deep neural networks have more power to express the functions that they attempt to approximate because of the combinatorial advantage of the composition of functions (Goodfellow et al., 2016). A crucial aspect of the development of the DINN is requiring the nonlinear activation function at each layer $F^{(\ell)}$ be monotonic and Lipschitz so that the final output can also be Lipschitz and as sharp as possible.

3.2. Embedding of Interval Input

The raw real-valued features X and targets Y in the training set are converted as needed to intervals $\mathbf{X}, \mathbf{Y}$ in an embedding layer using the specialized INTLAB toolbox in MATLAB (Rump, 1999), according to their upper and lower bounds. The interval embedding corresponds to layer 0 of the network. Eq. 2 is used in order to define the interval input.

3.3. Training

During training, we find a set of optimal parameters $\mathbf{W} \in \mathbb{IR}$ for each layer of the network by minimizing the loss function. In order to minimize the interval loss function $\mathcal{L}(\widehat{F}(\mathbf{X}_i), \mathbf{Y})$, we use first-order gradient based optimization. In particular, we use mini-batch stochastic gradient descent (SGD) and some of its variants, SGD with momentum (Polyak, 1964) and Adam (Kingma and Ba, 2014). The gradient descent update rule at step $k+1$ for sample i at layer l is defined as

$$\mathbf{W}_{k+1}^{(\ell)} = \mathbf{W}_{k}^{(\ell)} - \alpha_k \nabla_{\mathbf{W}_k}^{(\ell)} \mathcal{L}(\widehat{F}(\mathbf{X}_i), \mathbf{Y}_i), \tag{3}$$

$$\mathbf{b}_{k+1}^{(\ell)} = \mathbf{b}_{k}^{(\ell)} - \alpha_k \nabla_{\mathbf{b}_k}^{(\ell)} \mathcal{L}(\widehat{F}(\mathbf{X}_i), \mathbf{Y}_i), \tag{4}$$

where at each step k, $\mathbf{W}_k$ is the interval weight matrix , $\mathbf{b}_k$ is the interval bias vector, α_k is the step size, $\nabla_{\mathbf{W}_k}\mathcal{L}(\cdot)$ and $\nabla_{\mathbf{b}_k}\mathcal{L}(\cdot)$ are the gradients of the loss function with respect to parameters $\mathbf{W}_k$ and $\nabla_{\mathbf{b}}$. With mini-batch gradient descent, the algorithm chooses uniformly at random a batch of samples of size B from the full training set $\mathcal{T}$ and updates the gradient with Eqs. 3 and 4, which trains the network. There are three major steps to training a deep neural network:

1. Compute the loss at each training iteration via forward propagation of the input activations through the network's layers.

2. Compute the gradient of the loss function with respect to the model's parameters via the Backpropagation algorithm.

3. Minimize the loss function via stochastic gradient descent.

3.3.0.1. *Interval Forward Propagation* Algorithm 1 computes the predictions $\widehat{F}(\mathbf{X})$ and the loss $\mathcal{L}(\widehat{F}(\mathbf{X}_i), y)$ of the DINN through forward-propagation of each mini-batch of data. The layer output $\mathbf{z}^{(\ell)}$ (scores) is computed through an affine transformation as $\mathbf{z}^{(\ell)} = \mathbf{h}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)}$, at each layer ℓ; $\mathbf{W}^{(\ell)} \in \mathbb{IR}^{H^{(\ell-1)} \times H^{(\ell)}}$, $\mathbf{h}^{(\ell-1)} \in \mathbb{IR}^{B \times H^{(\ell-1)}}$, $\mathbf{b}^{(\ell)} \in \mathbb{IR}^{H^{(\ell)}}$, and $\mathbf{z}^{(\ell)} \in \mathbb{IR}^{B \times H^{(\ell)}}$, $H^{(\ell-1)}$ is the dimension of layer $\ell - 1$, $H^{(\ell)}$ is the dimension of layer ℓ, and B is the number of samples in each mini-batch of data. Then, the layer output $\mathbf{z}^{(\ell)}$ passes through an element-wise activation function $F(\mathbf{z}^{(\ell)})$. The default activation function for the DINN is a `ReLU`, defined as

$$\texttt{ReLU}(\mathbf{z}^{(\ell)}) = \max(\mathbf{z}^{(\ell)}, 0). \tag{5}$$

In the regression setting, for the final layer L, the prediction is simply the affine transformation of final output $\mathbf{z}^{(L)}$, without a `ReLU`. The loss (i.e., error) at the end of each epoch is computed using Mean Squared Error (MSE) for each mini-batch of size B, as

$$\mathcal{L}(\widehat{F}(\mathbf{X}), \mathbf{Y}) = \frac{1}{2B} \sum_{i=1}^{B} (\widehat{F}(\mathbf{X}_i) - \mathbf{Y}_i)^2. \tag{6}$$

The loss is an interval scalar. Notice that `ReLU` is a monotonically increasing function, which allows for calculation of sharp interval enclosures. The `ReLU` is the standard activation function for the DINN but it can be replaced by a different activation function with ease.

Algorithm 1: Forward Propagation Algorithm for Each Mini-batch of Data

Obtain the input raw real-valued features X and targets Y or labels y
Embed X into interval input features $\mathbf{X}$ considering lower $\underline{X}$ and upper bounds $\overline{X}$ for each sample
Embed Y into interval input features $\mathbf{Y}$ considering lower $\underline{Y}$ and upper bounds $\overline{Y}$ for each sample
Choose number of layers L
Require model parameters $\mathbf{W}^{(\ell)}$, for each layer ℓ
$\mathbf{h}^{(0)} = \mathbf{X}$
for $\ell = 1 : L$ **do**
  $\mathbf{z}^{(\ell)} = \mathbf{h}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)}$
  $\mathbf{h}^{(\ell)} = F(\mathbf{z}^{(\ell)})$
Prediction $\widehat{F}(\mathbf{X}) = \mathbf{h}^{(L)}$
Compute loss $\mathcal{L}(\widehat{F}(\mathbf{X}_i), \mathbf{Y}_i)$

3.3.0.2. *Interval Backpropagation* The first step for SGD is to find the gradients $\nabla_{\mathbf{W}}\mathcal{L}$. These gradients are computed via automatic differentiation in the computational graph of the neural network via Backpropagation. For the DINN, we use the Backpropagation algorithm to take derivatives of interval matrices. In particular, the chain rule is used to compute the gradient as follows

$$\nabla_{\mathbf{W}^{(\ell)}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(\ell)}} \frac{\partial \mathbf{z}^{(\ell)}}{\partial \mathbf{W}^{(\ell)}} \tag{7}$$

where at layer ℓ, $\frac{\partial \mathbf{z}^{(\ell)}}{\partial \mathbf{W}^{(\ell)}}$ is the partial derivative of the score with respect to the weights, and $\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(\ell)}}$ is the partial derivative of the loss with respect to the score at layer ℓ.

This procedure is summarized in Algorithm 2 for general loss and activation functions. In the algorithm, the $\boldsymbol{\delta}$'s denote error terms that recursively update the error in the network at each layer. These interval gradients are accumulated in the $\mathbf{G}$ data structure in the code. Regularization $\alpha\Omega(\mathbf{W})$ is also added to the gradient calculation and explained in the next section. The algorithm requires only an interval subtraction, an interval matrix-vector multiplication, and updating the $\mathbf{G}$ data structure for each mini-batch of data.

Algorithm 2: Backpropagation Algorithm for Interval Loss for Each Mini-batch of Data

Run forward computation algorithm to obtain layer activations $\mathbf{h}^{(\ell)}$
Compute gradient of loss at output layer $\boldsymbol{\delta}_{out}^{(L)}$
$\mathbf{G} = \boldsymbol{\delta}_{out}^{(L)}$
$\nabla_{\mathbf{W}}^{(L)}\mathcal{L} = \mathbf{h}^{(L-1)^T}\mathbf{G} + \lambda\nabla_{\mathbf{W}}^{(L)}\Omega(\mathbf{W})$
$\nabla_{\mathbf{b}}^{(L)}\mathcal{L} = \mathbf{G} + \lambda\nabla_{\mathbf{b}}^{(L)}\Omega(\mathbf{W})$
$\mathbf{G} \leftarrow \mathbf{G}\mathbf{W}^{(L)^T}$
for $\ell = L-1, \ldots, 1$ *layers* **do**
  $\mathbf{G} \leftarrow \boldsymbol{\delta}_{\mathbf{z}}^{(\ell)} = \mathbf{G} \odot F'(\mathbf{z}^{(\ell)})$
  $\nabla_{\mathbf{W}}^{(\ell)}\mathcal{L} = \mathbf{h}^{(\ell-1)^T}\mathbf{G} + \lambda\nabla_{\mathbf{W}}^{(\ell)}\Omega(\mathbf{W})$
  $\nabla_{\mathbf{b}}^{(\ell)}\mathcal{L} = \mathbf{G} + \lambda\nabla_{\mathbf{b}^{(\ell)}}\Omega(\mathbf{W})$
  $\mathbf{G} \leftarrow \mathbf{G}\mathbf{W}^{(\ell)^T}$

For the DINN as presented on this paper, with MSE loss and `ReLU` activations, the partial derivatives in Algorithm 2 specialize as follows. At the final layer L we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(L)}} = \boldsymbol{\delta}_{\mathbf{z}}^{(L)} = \boldsymbol{\delta}_{out}^{(L)} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}}\frac{\partial \mathbf{h}^{(L)}}{\partial \mathbf{z}^{(L)}} = (\mathbf{h}^{(L)} - \mathbf{Y}). \tag{8}$$

For $\frac{\partial \mathbf{z}^{(\ell)}}{\partial \mathbf{W}^{(\ell)}}$, we have

$$\frac{\partial \mathbf{z}^{(\ell)}}{\partial \mathbf{W}^{(\ell)}} = \mathbf{h}^{(\ell-1)^T}. \tag{9}$$

At the hidden layers ($\ell = L-1, L-2, \ldots, 1$), going backwards in the network graph, the partial derivative with respect to the score is

$$\begin{aligned}\boldsymbol{\delta}_{\mathbf{z}}^{(\ell)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(\ell)}} &= \boldsymbol{\delta}_{\mathbf{z}}^{(\ell+1)}\mathbf{W}^{(\ell+1)^T} \odot \underbrace{F'(\mathbf{z}^{(\ell)})}_{\substack{\text{derivative of activation}}} \\ &= \boldsymbol{\delta}_{\mathbf{z}}^{(\ell+1)}\mathbf{W}^{(\ell+1)^T} \odot \underbrace{\mathbb{1}\{\mathbf{h}^{(\ell)} \geq 0\}}_{\substack{\text{derivative of } \texttt{ReLU} \\ = 1 \text{ if } \mathbf{h}^{(\ell)} \geq 0 \\ = 0 \text{ otherwise.}}}.\end{aligned} \tag{10}$$

where $\mathbf{A} \odot \mathbf{B}$ is the Hadamard (element-wise) product of $\mathbf{A}$ and $\mathbf{B}$, and $\mathbb{1}\{\cdot\}$ is the indicator function.

Finally, the gradient with respect to the weights at each layer is (Eq. 7)

$$\nabla_{\mathbf{W}^{(\ell)}}\mathcal{L} = \mathbf{h}^{(\ell-1)^T}\boldsymbol{\delta}_{\mathbf{z}}^{(\ell)}, \tag{11}$$

and the gradient with respect to the biases at each layer is

$$\nabla_{\mathbf{b}^{(\ell)}}\mathcal{L} = \boldsymbol{\delta}_{\mathbf{z}}^{(\ell)}. \tag{12}$$

3.3.0.3. *Regularization* Regularization is used for two purposes in the DINN. First, regularization is used to prevent overfitting to the training set, with the goal of achieving better performance in unseen data. It achieves so by shrinking the weight contribution of less relevant features. Second, it stabilizes the solution which for finding interval gradients is of crucial importance. Two forms of regularization are used for the DINN. $L1$ (Lasso) regularization uses the l_1, as

$$\Omega(\mathbf{W}) = \|\mathbf{W}\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |\mathbf{A}_{ij}|, \tag{13}$$

and $L2$ (Ridge) regularization which is defined as

$$\Omega(\mathbf{W}) = \sum_{m} \sum_{n} \mathbf{W}_{m,n}^2. \tag{14}$$

The $L2$ regularization is expressed in a way to directly square the elements of the $\mathbf{W}$ in INTLAB, so as to obtain a monotonic expression. The regularization is added to the data loss, as $J(\widehat{F}(\mathbf{X}), \mathbf{Y}; \mathbf{W}) = \mathcal{L}(\widehat{F}(\mathbf{X}), \mathbf{Y}; \mathbf{W}) + \lambda \cdot \Omega(\mathbf{W})$, where $J(\widehat{F}(\mathbf{X}), \mathbf{Y}; \mathbf{W})$ is the regularized (total) loss and λ is the regularization strength.

4. Supervised Interval Field

At inference time, we query the trained DINN model with new interval data and obtain interval predictions using Eq. 1. These predictions are in turn used to develop the interval field, here named Supervised Interval Field (SIF), by mapping the DINN prediction at each spatial coordinate in the IFEM mesh. Fig. 3 shows the overall process of the DINN in a forward problem. We previously presented a SIF (Betancourt et al., 2018) for the spatiotemporal variation of IFEM input variables though a real-valued deep neural network. In the this work, with the DINN we add the capability to process the interval uncertainty in the domain in addition to the spatiotemporal variation.

4.1. SIF discretization for IFEM

The SIF predictions are independent of the IFEM mesh. Thus a post-processing discretization of the SIF to map to the elements in the mesh of the IFEM is required. For each element in the FE mesh, the average of the SIF predictions covered by the corresponding physical area of the element can be taken. After the discretization, a conventional IFEM analysis can be performed.

5. Experiments

5.1. Concrete Strength Dataset

The concrete strength dataset was retrieved from the UCI machine learning repository (Dua and Graff, 2017) and originated in (Yeh, 1998b). It consists of 1030 samples, each with eight concrete mix features and their corresponding cylinder compressive strength. The eight features are

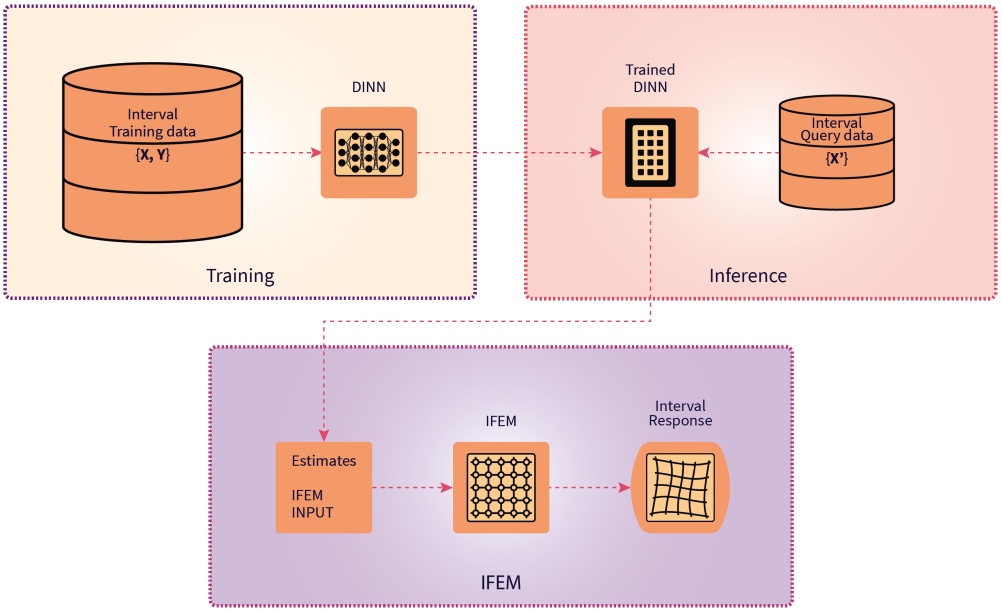

Figure 3. Overall process for forward problem consisting of three stages. **Training**: train the DINN with interval data. **Inference**: Then, make predictions with trained DINN at inference time with new data. **IFEM**:Use the predictions of DINN to form the SIF and perform the IFEM analysis. Finally, get the interval response from the IFEM.

$\mathbf{X} = \{$`Cement, Slag, Fly Ash, Water, Superplasticizer (SP), Coarse Aggregate (CA), Fine Aggregate (FA), Age`$\}$, while the continuous-valued target is $\mathbf{Y} = \{$`Compressive Strength`$\}$. Details about the experiments used to gather the data are included in (Yeh, 1998b). Fig. 4 shows the quartiles of the data and its pair-wise correlations.

Previous works have studied the predictive power of neural networks for this dataset (Yeh, 1998a) (Yeh, 1999), (Yeh, 2003), (Yeh, 2006), (Gal and Ghahramani, 2016). Two important factors area notable in this dataset. First, the data is aggregated from multiple sources, many of which were from experiments done in the 1970s and tested for high strength concrete as defined at the time—which was usually around 40 MPa (5800 psi). Today, high strength concrete is defined by the ACI (363, 2005) as compressive strengths above 55 MPa (8000 psi). Thus, care must be used if predicting concrete strength with *new* high strength data using a trained machine learning model using this dataset. Second, as mentioned in (Yeh, 1998a), the dataset contains uncertainty from experiments being conducted to different standards (i.e., different cylinder sizes), missing features (e.g. Fly Ash content missing for many samples), unspecified type of superplasticizers, unspecified curing temperatures, and lack of knowledge of precision of measurement devices.

The DINN model presented here can have multiple applications from an optimization perspective. However, two very important application can be readily derived from the DINN model. The first, is to use it in a forward problem, to estimate the modulus of elasticity of the structure from

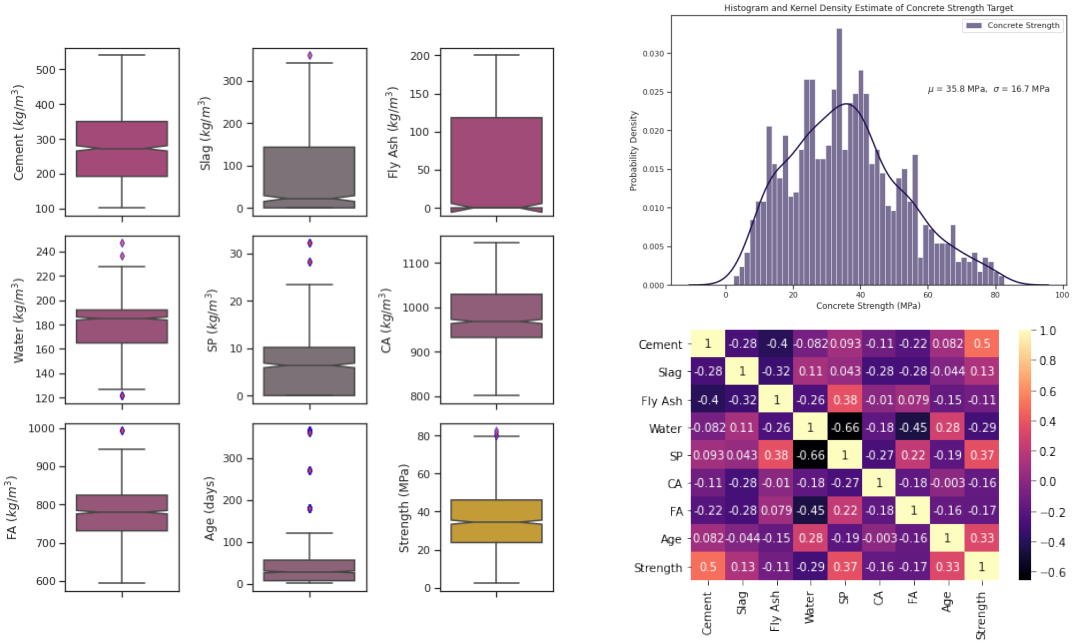

Figure 4. Concrete Strength Dataset Boxplots (left); Concrete Strength (Target) Probability Density (top right); Concrete Strength Dataset feature correlations heatmap (bottom right);

only its mix components. The second, is to design more optimal concrete mixes to achieve structural design concrete strength while decreasing cement content.

5.1.1. *Experimental Details*
The experiments consists of the followings steps (see Fig. 3 for overall process diagram):

1. Perform interval uncertainty quantification on the dataset and convert real-valued data to interval-valued data based on the determined uncertainty levels. This step produces the interval training data (Sec. 3.2).

2. Train the DINN with the interval training data (Sec. 3.3).

3. Perform inference with the trained DINN by querying it with new interval data to obtain the corresponding target predictions (i.e., concrete strength). The results of this query forms the *interval field* which is used as input to the IFEM (Sec. 4).

Once the interval field SIF is computed with the DINN and discretized to map to the IFEM mesh (Sec. 4), we can perform an IFEM analysis to find structural responses.

5.1.1.1. *Interval Uncertainty* Overall, the uncertainty in this dataset is largely epistemic for the reasons given in 5.1. Thus, we use intervals to represent the input uncertainty and propagate it through the DINN by using the procedure in Sec. 3. We use an uncertainty level of $\beta = 15\%$ for

the superplasticizer and $\beta = 5\%$ for all other features in $\mathbf{X}$. We have one set-up with degenerate intervals for the targets $\mathbf{Y}$ (concrete strength), which implies that we have uncertainty in the features but not in the targets.

5.1.1.2. *DINN Model Details* We trained the DINN with SGD and SGD with momentum. We use three hidden layers, 500 units per layer, and 200 epochs. We use randomized search cross-validation (Bergstra and Bengio, 2012) with 10 folds to find optimal hyperparameters for the DINN.

5.1.1.3. *Results* Fig. 5 shows the training loss and the test set predictions. It can be seen that the predictions of the DINN bound the ground truth targets for most test set samples. Additionally, it is observed that the DINN propagated the uncertainty in the inputs without producing overly wide interval predictions.

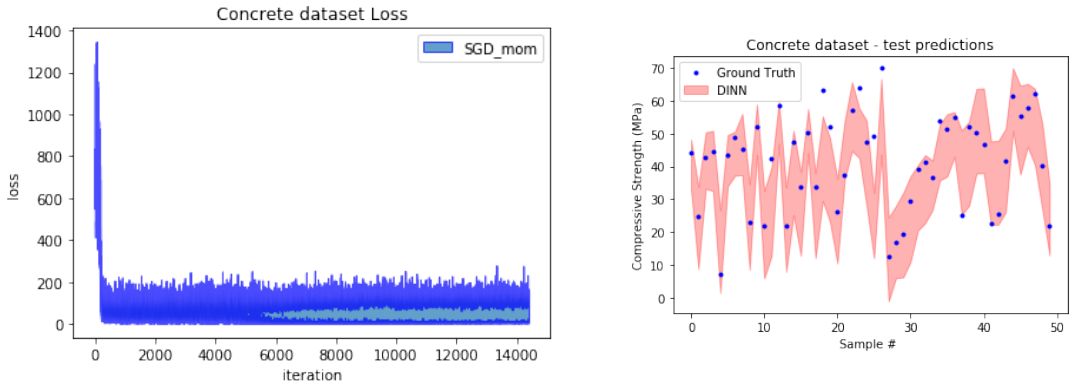

Figure 5. Training loss history (left); Testing Set predictions (right)

6. Conclusion

We have presented a novel interval deep learning algorithm called the Deep Interval Neural Network (DINN), which is capable to make predictions with interval-valued data. Using the DINN, we have presented a method, the SIF, to obtain an interval field from indirect measurements under uncertainty for IFEM input properties.

Our method gains advantage over frequentists or Bayesian methods when we do not have enough information about the uncertainties. It is especially useful in first-of-a-kind safety-critical structural mechanics applications, granted that a data collection program is put into practice.

References

363, A. C. (2005). High-strength concrete. *American Concrete Institute–Special Publication*, 228.

Adam, S. P., Magoulas, G. D., Karras, D. A., and Vrahatis, M. N. (2016). Bounding the search space for global optimization of neural networks learning error: an interval analysis approach. *The Journal of Machine Learning Research*, 17(1):5898–5937.

Ben-Tal, A. and Nemirovski, A. (2002). On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. *SIAM Journal on Optimization*, 12(3):811–833.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Betancourt, D., Muhanna, R., and Mullen, R. (2018). Interval field for spatially and temporally dependent uncertainty–machine learning approach. In *Proceedings of the 8th international workshop on reliable engineering computing*, volume 8. University of Liverpool.

Czarnecki, W. M. and Podolak, I. T. (2013). Machine learning with known input data uncertainty measure. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 379–388. Springer.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Ferson, S., Kreinovich, V., Hajagos, J., Oberkampf, W., and Ginzburg, L. (2007). Experimental uncertainty estimation and statistics for data having interval uncertainty. *Sandia National Laboratories, Report SAND2007-0939*, 162.

Freitag, S., Graf, W., and Kaliske, M. (2011). Recurrent neural networks for fuzzy data. *Integrated Computer-Aided Engineering*, 18(3):265–280.

Freitag, S., Muhanna, R. L., and Graf, W. (2012). A particle swarm optimization approach for training artificial neural networks with uncertain data. In *Proceedings of the 5th International Conference on Reliable Engineering Computing, Litera, Brno*, volume 151.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Ghanem, R. G. and Spanos, P. D. (2003). *Stochastic finite elements: a spectral approach*. Courier Corporation.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Hauseux, P., Hale, J. S., Cotin, S., and Bordas, S. P. (2018). Quantifying the uncertainty in a hyperelastic soft tissue model with stochastic parameters. *Applied Mathematical Modelling*, 62:86–102.

Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

He, Q. and Chen, J.-S. (2020). A physics-constrained data-driven approach based on locally convex reconstruction for noisy database. *Computer Methods in Applied Mechanics and Engineering*, 363:112791.

Hernandez, C., Espf, J., Nakayama, K., and Fernandez, M. (1993). Interval arithmetic backpropagation. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 1, pages 375–378. IEEE.

Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869.

Huang, D., Fuhg, J. N., Weißenfels, C., and Wriggers, P. (2020). A machine learning based plasticity model using proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 365:113008.

Imholz, M., Faes, M., Cerneels, J., Vandepitte, D., and Moens, D. (2016). On the comparison of two novel interval field formulations for the representation of spatial uncertainty. In *Proceedings of the 7th international workshop on reliable engineering computing*, volume 7, pages 367–378. University of Bochum.

Ishibuchi, H. and Tanaka, H. (1991). An extension of the bp-algorithm to interval input vectors-learning from numerical data and expert's knowledge. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1588–1593. IEEE.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Korzeniowski, T. F., Reppel, T., and Weinberg, K. (2019). A juxtaposition of data driven and stochastic finite element analyses for problems with noisy material data. *PAMM*, 19(1):e201900197.

Kreinovich, V., Lakeyev, A. V., Rohn, J., and Kahl, P. (2013). *Computational complexity and feasibility of data processing and interval computations*, volume 10. Springer Science & Business Media.

Liu, C., Arnon, T., Lazarus, C., Barrett, C., and Kochenderfer, M. J. (2019). Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*.

McDermott, P. L. and Wikle, C. K. (2019). Bayesian recurrent neural network models for forecasting and quantifying uncertainty in spatial-temporal data. *Entropy*, 21(2):184.

Moens, D. and Hanss, M. (2011). Non-probabilistic finite element analysis for parametric uncertainty treatment in applied mechanics: Recent advances. *Finite Elements in Analysis and Design*, 47(1):4–16.

Moens, D., Munck, M., Desmet, W., and Vandepitte, D. (2011). Numerical dynamic analysis of uncertain mechanical structures based on interval fields. In *IUTAM symposium on the vibration analysis of structures with uncertainties*, pages 71–83. Springer.

Moore, R. E., Kearfott, R. B., and Cloud, M. J. (2009). *Introduction to interval analysis*, volume 110. Siam.

Muhanna, R. L. and Mullen, R. L. (2001). Uncertainty in mechanics problems—interval–based approach. *Journal of Engineering Mechanics*, 127(6):557–566.

Neumaier, A. (1990). *Interval methods for systems of equations*, volume 37. Cambridge university press.

Nuttall, J. (2018). Estimating spatial correlations from cpt data using neural networks and random fields. In *9th European conference on numerical methods in geotechnical engineering (NUMGE2018), At Porto, Portugal*.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

Reynders, E. and De Roeck, G. (2014). Vibration-based damage identification: the z24 benchmark.

Rump, S. M. (1999). Intlab—interval laboratory. In *Developments in reliable computing*, pages 77–104. Springer.

Sofi, A. (2015). Structural response variability under spatially dependent uncertainty: stochastic versus interval model. *Probabilistic Engineering Mechanics*, 42:78–86.

Sofi, A. (2017). Euler–bernoulli interval finite element with spatially varying uncertain properties. *Acta Mechanica*, pages 1–17.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Verhaeghe, W., Desmet, W., Vandepitte, D., and Moens, D. (2011). Uncertainty assessment in random field representations: an interval approach. In *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, pages 1–6. IEEE.

Walley, P. (1991). *Statistical reasoning with imprecise probabilities*. Chapman & Hall.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018). Formal security analysis of neural networks using symbolic intervals. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1599–1614.

Wu, D. and Gao, W. (2017). Hybrid uncertain static analysis with random and interval fields. *Computer Methods in Applied Mechanics and Engineering*, 315:222–246.

Yang, D. and Liu, Y. (2018). L1/2 regularization learning for smoothing interval neural networks: Algorithms and convergence analysis. *Neurocomputing*, 272:122–129.

Yang, D. and Wu, W. (2012). A smoothing interval neural network. *Discrete Dynamics in Nature and Society*, 2012.

Yang, Z., Lin, D. K., and Zhang, A. (2019). Interval-valued data prediction via regularized artificial neural network. *Neurocomputing*, 331:336–345.

Yeh, I.-C. (1998a). Modeling concrete strength with augment-neuron networks. *Journal of Materials in Civil Engineering*, 10(4):263–268.

Yeh, I.-C. (1998b). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808.

Yeh, I.-C. (1999). Design of high-performance concrete mixture using neural networks and nonlinear programming. *Journal of Computing in Civil Engineering*, 13(1):36–42.

Yeh, I.-C. (2003). Prediction of strength of fly ash and slag concrete by the use of artificial neural networks. *J. Chin. Inst. Civil Hydraul. Eng*, 15(4):659–663.

Yeh, I.-C. (2006). Analysis of strength of concrete using design of experiments and neural networks. *Journal of Materials in Civil Engineering*, 18(4):597–604.

Zhang, H. (2005). *Nondeterministic linear static finite element analysis: an interval approach*. PhD thesis, Georgia Institute of Technology.