buildings

MDPI

Article

Imperfection Sensitivity Detection in Pultruded Columns Using Machine Learning and Synthetic Data

Michail Tzimas [1,†] and Ever J. Barbero [2,*,†]

[1] Independent Researcher, Morgantown, WV 26505, USA; mtzimas92@gmail.com
[2] Department of Mechanical, Materials and Aerospace Engineering, West Virginia University, Morgantown, WV 26506, USA
* Correspondence: ejbarbero@mail.wvu.edu
† These authors contributed equally to this work.

Abstract: Experimental and theoretical solutions have shown that imperfections in wide-flanged structural columns may reduce the failure load of the column by as much as 30% with respect to that of a perfect column. Therefore, the early detection and prevention of such imperfections, which would likely reduce the load capacity of a structure, are critical for avoiding catastrophic failure. In the present article, we show how machine learning may be used to detect imperfection sensitivity in pultruded columns using observable column deformations occurring at loads as low as 30% of the design load. Abaqus simulations were used to capture the behavior of such columns of various lengths under service load. The deformations found from the simulations were used to train the machine learning algorithm. Similar deformations could be easily collected from in-service columns using inexpensive instrumentation. With over 3000 test cases, 95% accuracy in the correct detection of imperfection sensitivity was found. We anticipate that the proposed machine learning pipeline will enhance structural health monitoring, providing timely warning for potentially compromised structures.

Keywords: structural health monitoring; machine learning; buckling; imperfection sensitivity; failure prevention

Citation: Tzimas, M.; Barbero, E.J. Imperfection Sensitivity Detection in Pultruded Columns Using Machine Learning and Synthetic Data. *Buildings* **2024**, *14*, 1128. https://doi.org/10.3390/buildings14041128

Academic Editors: Tadeh Zirakian and David M. Boyajian

Received: 21 March 2024
Revised: 3 April 2024
Accepted: 16 April 2024
Published: 17 April 2024

1. Introduction

In this work, we developed a method to detect imperfection sensitivity in columns using inexpensive instrumentation, regardless of the source of deterioration. For example, a column in a parking garage could sustain imperceptible damage from a vehicle impact or experience polymer degradation due to aging. Since imperfection sensitivity can lead to catastrophic compression failure without warning, our proposed work is relevant to civil infrastructure, particularly in earthquake scenarios [1].

Regardless of the deterioration source, if we detect a deviation in deformation from the expected service level, our machine learning (ML) algorithm can identify imperfection sensitivity in the column, leading to a reduction in the failure load. This occurs without needing to identify the specific material or geometric deterioration causing the deformation anomaly. Although the ML algorithm includes the source of material or geometric deterioration in its training, the damage detection instrumentation does not need to identify this source. Instead, it detects the aggregate effect of damage using simple techniques, such as measuring lateral deflection.

In this work, we did not need to physically test columns for failure because the accuracy of the simulation software, i.e., ABAQUS 2020® [2], is widely trusted for elastic analysis of composite materials and structures. Furthermore, Abaqus has demonstrated good agreement with experimental data for materials and structures similar to those in our study [3].

With the simulation software thus validated, we generated synthetic data by simulating thousands of cases covering a spectrum of loads, dimensions, and materials as well as types, magnitudes, and localization of damage. Then, we used the synthetic data to train an ML algorithm. Finally, we proved that the trained algorithm is accurate in predicting the failure load of simulated cases that we did not use for training.

To keep the scope of work manageable with our time and resource limitations, in this work, we only developed synthetic data including a spectrum of (a) loads, (b) magnitude of pre-existing imperfection amplitude, and (c) column dimensions, while (d) material properties, (e) other types of damage, and (f) localization of damage, were relegated to further studies.

In practice, we envision inexpensive instrumentation, such as a laser deflectometer continuously monitoring the deflection of a column that has been identified as critical to the structure's survivability in case of an earthquake. With that simple deflection input, ML can predict the failure load deterioration in real time, before the earthquake occurs. If a pre-established threshold of predicted failure load reduction is reached, a warning would be triggered, prompting remedial actions to prevent structural overload.

We have not attempted to implement instrumentation to validate deflection measurement techniques, as we have confidence in the accuracy and affordability of methods such as laser deflectometry, digital image correlation, and photogrammetry [4].

Our proposed method requires only four lateral deflections to be monitored to extract Fourier coefficients. However, increasing the number of measured deflections will enhance the performance of the method. Thus, burdensome problems with large amounts of data transmission and processing that might be encountered in classical structural health monitoring (SHM) approaches are eliminated [4].

The proposed methodology requires the ML algorithm to be trained with synthetic data generated by simulations that include a spectrum of all possible types of damage, magnitudes of damage, and localization of damage. This spectrum of data is a surrogate for the last three out of the four classical SHM concerns, namely (1) damage detection, (2) qualification, (3) quantification, and (4) localization.

In our approach, we trained an ML algorithm to predict *the effect of damage*, rather than detecting damage itself. If the effects of damage can be elucidated from in-service deformations or any other response that can be easily measured, cumbersome damage detection is avoided. By utilizing a discrete set of Euler and local mode lateral deflections, we accurately predict imperfection sensitivity in columns, leading to an effective prediction of failure load reduction. We discuss our methodology and data acquisition in Section 2, present results in Section 3, and provide conclusions and future directions in Section 4.

2. Materials and Methods

2.1. Case Study of a Pultruded Column: Finite Element Simulation Data Acquisition

Pultruded structural shapes are thin-walled to take advantage of the high compressive strength of the fiber-reinforced composite material and to remediate the relatively low modulus of elasticity of the material. The flanges of the thin-walled section buckle first for stubby columns, but as the slenderness λ increases, Euler buckling occurs [5]. The local buckling load P_L is relatively constant, but the Euler buckling load, P_E, decreases sharply with the slenderness of the column. At the critical slenderness λ_{cr}, local and Euler loads coincide, i.e., $P_{cr} = P_E = P_L$ [6].

The slenderness of the column can be used in parametric studies. For perfect columns, there are two (isolated) observable modes, the local mode, for $\lambda \leq 1$, $P_{max} = P_{cr}$, and the Euler mode, for $\lambda \geq 1$, $P_{max} = (\pi^2 EI)/(KL)^2$, where E, I, are material properties, L is the length, and K depends on the type of end-supports of the column.

Real columns are not perfect but rather have imperfections, which may be internal or external to the column. External imperfections include uneven axial lengths and eccentricities or non-uniformity of the applied load. Internal imperfections can be caused by damage or aging of the material. Therefore, for an imperfect column, the buckling load is less than

the load predicted by either of the isolated modes described above, as shown in [3,6,7] and multiple citations therein. Imperfection sensitivity can be reduced with a combination of manufacturing processes and targeted modeling as shown in ref. [8].

In Figure 1, both local and Euler modes are shown; however, for slenderness λ between 0.7 and 1.2, multiple data points are observed to fall outside the isolated responses. Upon closer examination, these data points reveal the interaction between local and Euler modes. A design equation has been proposed that captures the behavior of the experimental observations across the spectrum of λ [3,6].

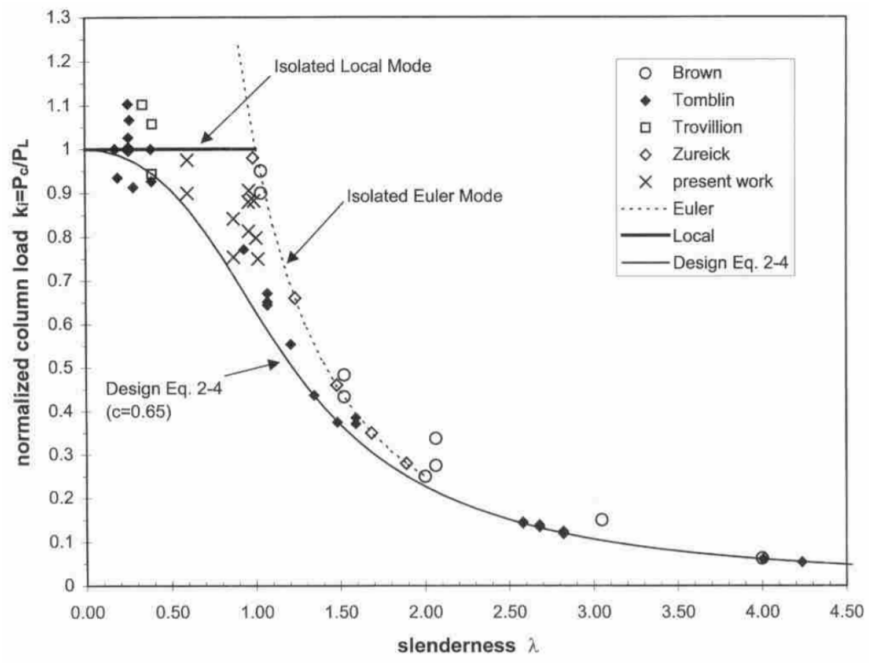

Figure 1. Column experimental data and prediction by various equations used in design [9].

To simulate imperfection-sensitive columns, Finite Element Analysis (FEA) is employed. Abaqus software is utilized to model columns with specified dimensions and material properties. Material properties are given in Tables 1 and 2. Additional information regarding calculation and use of these material properties in Abaqus can be found in refs. [10] and Ex. 3.11 and 4.4 in [11].

Table 1. Material properties. Transverse shear coefficients used in ABAQUS simulations. Units for both flange and web: MPa mm.

Transverse Shear	K11	K12	K22
Flange	15,788	0	15,338
Web	16,378	0	15,955

Table 3 shows information regarding the FEA parameters used for meshing and boundary conditions. One end of the column has a symmetric boundary condition to reduce the overall length of the model. The cross-section, on the other end, has displacement constraints for rigid body motion, tied to a reference point. The load of the simulation is applied on the same reference point as a compressive force.

Table 2. Material properties. Upper symmetric coefficients of the General Shell Stiffness Matrix used in Abaqus simulations. Units for both flange and web, rows 1–3: MPa mm, rows 4–6: MPa mm^3.

Flange	163,370	31,996	0	0	0	0
	0	87,165	0	0	0	0
	0	0	25,649	0	0	0
	0	0	0	489,226	116,521	0
	0	0	0	0	308,006	0
	0	0	0	0	0	91,080
Web	158,176	32,038	0	0	0	0
	0	88,103	0	0	0	0
	0	0	26,132	0	0	0
	0	0	0	767,573	182,832	0
	0	0	0	0	420,500	0
	0	0	0	0	0	124,985

Table 3. ABAQUS properties for FEA model.

Number of Elements	Length: 30	Width, Height: 4
Element Type	Shell, Quadratic, 6 DOF	S8R
Boundary Condition 1	Symmetry, ZSYMM	On one end
Boundary Condition 2	Dispacement, U1, U2, UR3	On reference point
Load	Concentrated Force, CF3	On reference point

The perfect column is defined as a fiber-reinforced plastic (FRP) beam with a wide flange (WF) with dimensions 6″ × 6″ × 3/8″ (WF 6×6). The width and height of the column are 6″, while the flange and web thickness are 3/8″ (9.525 mm). The length of the column is variable. Figure 2 shows the relevant dimensions.

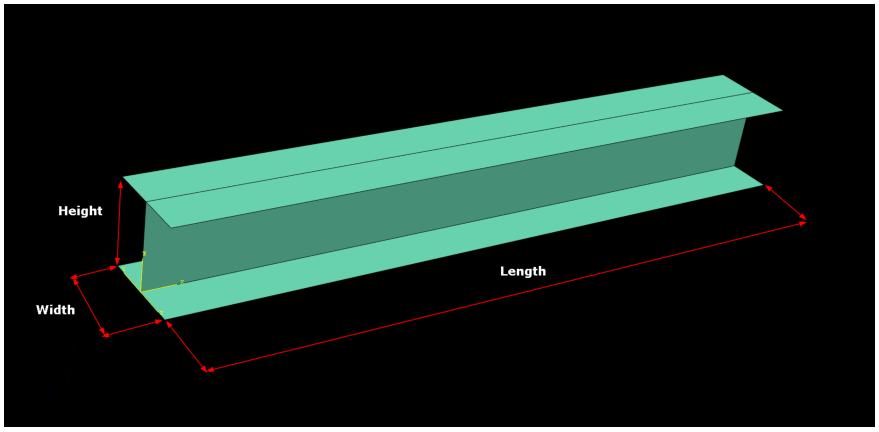

Figure 2. Column model in Abaqus.

In practice, these columns are typically produced by pultrusion. The material properties reported in the tables were obtained analytically using the properties of the matrix (vinyl ester) and fibers (E-glass). Calculated material properties were validated experimentally [12]. Furthermore, fiber density, architecture, and placement within the cross-sections were used in the calculations. The relevant matrix and fiber properties are widely available, while fiber architecture is proprietary to the manufacturer.

The critical length is found to be L_{cr} = 2280 mm, and the critical load is P_{cr} = 169.752 MPa. In order to generate multiple test cases, the slenderness λ was varied from 0.5 to 1.5 (equivalent lengths $L = 1140$ mm $- L = 3420$ mm) to cover most of the cases that fall within the region of interest, as shown in Figure 1.

Moreover, the chosen range of λ allows for cases that are expected to produce mode interaction [3], which is helpful in identifying whether columns are imperfection-sensitive

(IS) or not. Imperfection sensitivity (IS) is identified by the presence of mode interaction, characterized by both lateral Euler deflection and flange deformation. Columns exhibiting only local, or only Euler modes are classified as non-imperfection-sensitive (NIS). Each mode has some expected responses, with local mode showing deformation on the flanges (wave patterns) and Euler mode showing lateral deflection of the whole column. Therefore, to correctly identify IS columns, both lateral Euler deflection and flange deformation must be observed.

In FEA, the effects of imperfections can be modeled using perturbations in the geometry. In Figure 3, we provide typical results of a column under buckling conditions. In Figure 3a, we have flange deformation (wave undulations) for local mode, and in Figure 3c we have lateral deflection, i.e., Euler mode. Based on the above discussion, Figure 3a,c behave as NIS columns. On the other hand, Figure 3b depicts an IS column, since there is an observable lateral deflection coupled with flange deformation.

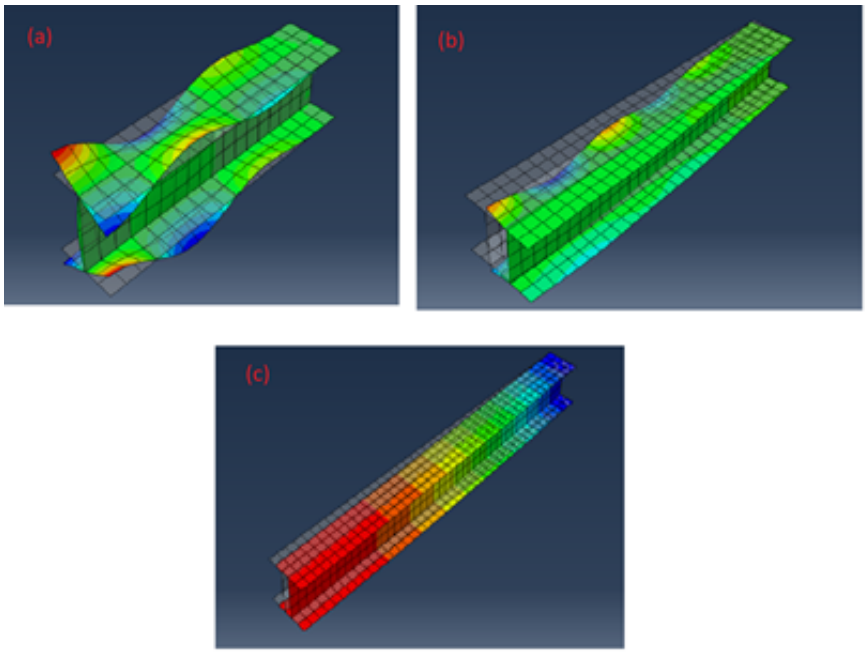

Figure 3. Abaqus simulation results for an FRP beam with wide flange of varying lengths: (**a**) when $\lambda << 1$, the deformation is limited to the flange; (**b**) deformation caused by the interaction of Euler and local modes; (**c**) when $\lambda >> 1$, Euler mode causes lateral deflection without flange undulations. The color scheme represents the magnitude of deformation as reported by Abaqus.

In this work, synthetic data generation involved two phases: simulating perfect columns and introducing imperfections. ABAQUS performed buckling analysis to obtain eigenvalues and eigenmodes, representing limit loads and deformation shapes, respectively. Imperfections are introduced based on combinations of eigenmodes, simulating realistic conditions. The process for producing the synthetic data was fully automated and involved a single buckling analysis per length and multiple runs of each individual length when modeling imperfections on a second pass.

For instance, in Figure 3c, depicting lateral deflection caused by the Euler mode, Abaqus illustrates the first eigenmode of the column, where the displacement is represented by the color scheme. Eigenmode 2 for the same column shows undulations and no lateral deflection. Thus, Abaqus essentially provides a Fourier transform of the leading components that make up all possible deformations of the column, since deformation in multiple axes and with different combinations of the Fourier modes is possible.

Imperfections were introduced based on combinations of two eigenmodes identified in the initial buckling simulations conducted for each length. The first eigenmode was always chosen to be the mode that produces lateral deflection (Euler). The second eigenmode was

always chosen to be the mode that produces undulations on the flange, like Figure 3a, with the smallest load. Imperfection values were set to geometric imperfections on the FEA mesh. Non-linear geometric analysis (NGA) in Abaqus is used to plot the load-deflection chart of the imperfect column.

Non-linear geometric analysis, utilizing the Riks method [13] in ABAQUS, is the second step of the buckling analysis to simulate realistic conditions. After NGA is concluded, results similar to Figure 3 can be found. Similar scenarios were observed in both experimental setups [14] and in simulations [9]. In this article, we conducted multiple Abaqus simulations, totaling N = 3750, aimed at detecting imperfection sensitivity through machine learning methods.

In this section, we discussed the methodology for acquiring data through Finite Element Analysis simulations of pultruded columns. The simulation setup includes modeling perfect columns and introducing imperfections to simulate real-world conditions. The parameters considered cover a range of scenarios, including mode interaction, to identify imperfection sensitivity. These simulations serve as the basis for generating synthetic data for subsequent analysis.

2.2. Machine Learning Model

Machine learning (ML) [15] is a branch of Artificial Intelligence (AI) and is a rapidly growing field of study. ML involves algorithm development that allows computers to learn from data without explicit programming. ML is widely used in a variety of fields from everyday applications like image recognition [16] to engineering applications such as microstructural characterization and prediction of mechanical response of crystalline materials [17,18]. Furthermore, ML has been used for failure mode identification and strength prediction in columns [19–22].

Applications of ML are generally separated into three types: supervised, unsupervised, and reinforcement learning [23]. In this work, we focus on supervised learning, where algorithms learn from labeled data to make predictions on unseen data. Neural networks (NNs) [24], inspired by the human brain's structure, are commonly used for supervised learning tasks. Specifically, we employed deep neural networks [25], or multilayer perceptrons (MLPs), which consist of input, hidden, and output layers.

Neural or deep neural networks are specific algorithms that are modeled after human brain synapses, exploring possible linear permutations and connections between data points in a set. In neural networks, there is an input layer for the features of the dataset, followed by a layer of hidden units and a final output layer. Deep neural networks operate on the same principle but with multiple hidden layers between input and output. The input and hidden layers consist of multiple nodes, upon which mathematical operations are performed. Based on the result of the operations and whether that result can be "activated" via an activation function, or not, the nodes can be discarded, or the result can move to the next layer. When the information only travels forward (i.e., from the input layer to the hidden layers and then to the output layer), the networks are considered feedforward neural networks. When there are at least three layers (including input and output) in a feedforward NN, it is considered a multilayer perceptron (MLP).

As the process moves forward and multiple combinations are tested, the network reaches the output layer, which normally consists of 2 nodes in binary format (0 or 1) with a probabilistic outcome. For example, if in the final hidden layer the permutations and activation function give a result of 0.1, the result is binarized as (0.9, 0.1) for the output layer, meaning that there would be a 90% chance that the particular data point would belong to group 0. Once multiple data points have gone through the network (known as a batch), the next batch follows until all batches (or training data) have gone through (known as an epoch).

During a forward pass of a batch, the neural nodes are trained to learn useful permutations by applying biases and weights. At the end of the training of the batch, the

predicted outcomes in the output layer are tested against the known outputs to define a loss function [26].

The loss function is propagated backwards through the layers to optimize the weights in each node of each hidden layer. With the end of training of an ML model (i.e., after all epochs have finished), the weights are supposed to be optimized and can be further validated and tested. If the validation and testing phases give results with similar accuracy to those of the training step, the ML model can be deployed (i.e., the optimized weights can be applied to data with similar features). At this point, the ML algorithm would be ready to be deployed in the field, where it would be capable of predicting, for example, the failure load reduction for a given set of deformations measured in the field in real time.

In this work, we employed deep neural networks i.e., multilayer perceptrons (MLPs) with four hidden layers which were trained on a set of 2625 columns with 151 identifying features and then validated on a set of 1125 columns with the same number of features.

We employed the Tensorflow Python library [27]. The input layer includes the whole training dataset, which is then passed through the hidden layers. Each hidden layer progressively shortens the number of available neurons (nodes) until the binary output unit is reached. Each layer, except the output, uses the Rectified Linear Unit activation function [28] in each node to assess whether the neuron is important to the training, or not. The output layer employs the softmax activation function in each of the two nodes to convert the value to a probability distribution of the two possible outcomes. The training lasted 100 epochs, with a batch size of 100 columns in training.

2.3. Finite Element Simulations and Feature Selection

In a supervised ML problem, both the inputs and outputs need to be known. Therefore, prior to our ML training, we needed to determine which columns are considered imperfection-sensitive based on our Abaqus simulations. To do so, the following assumptions were made:

$$P_{max} < P_{cr} \;\&\; P_{final} < 85\%P_{max} \;\&\; U_{final} > U_{P_{max}} \implies IS \tag{1}$$

This means that if the peak load (P_{max}) found in NGA is less than the critical load (P_{cr}) and the final load in the simulation (P_{final}) is at least 15% less than the peak load, and furthermore the final load occurs at a higher displacement (U_{final}) than that of the peak load (U_{max}), then the column is imperfection-sensitive, as shown in Figure 4. Some columns may satisfy some parts of the equation and, in particular, the first part. However, if there is no observable load drop, the columns are not necessarily imperfection-sensitive but may just be following the Euler or local modes.

Lateral deflection in Figure 4 is found from a reference point (RP) in the ABAQUS discretization, and extracted with the following ABAQUS Script:

```
xyList = xyPlot.xyDataListFromField(
odb=odb, outputPosition=NODAL, variable=((
'CF', NODAL), ('U', NODAL), ), nodeSets=('SET-RP', ))
x0 = session.xyDataObjects['CF:CF3 PI: ASSEMBLY N: 1']
x1 = session.xyDataObjects['U:U3 PI: ASSEMBLY N: 1']
```

The script implements the Riks method, controlling both the load P and the $U3$ deflection of the RP. The RP is located at the point of load application. Figure 4 is made from "force-stroke" curves for every 100th sample column.

Once IS samples are identified, the next step is to identify which inputs can be used to train our algorithm. The inputs need to be experimentally tractable so that the proposed method applies to field data as well. Moreover, it is preferred that instrumentation to collect filed data is inexpensive; while ABAQUS gives us great latitude in choosing inputs to train our algorithm, our options for experimental data are rather limited. Specifically, we cannot use the peak load or any load close to the service load because we would need to experimentally measure deformations when the column has failed or is close to failure.

Thus, our goal is to train our ML algorithm with data collected for loads no larger than 30% of the service load for any given column. Furthermore, it must be noted that we can use any observable deformations that occur for smaller loads, both for training and for in situ monitoring of the deployed SHM system. The length of the column is also an observable variable.

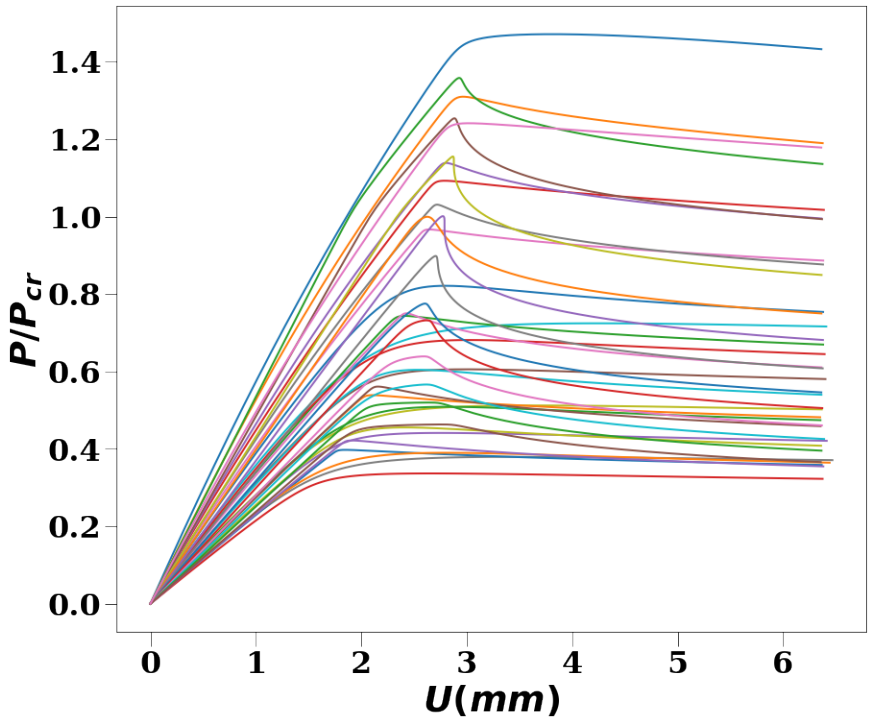

Figure 4. Normalized load vs. lateral deflection for every 100th column.

As imperfection sensitivity is the combination of local and Euler modes, and as local modes manifest themselves with undulations on the flanges of columns, we can use the undulations as our experimentally observable features. Undulations in local modes can be visually observed in loads as low as 20–30% of service loads, as corroborated by simulation presented herein and by experimental observations [14]; thus, they are prime candidates for use as inputs in our ML pipeline. Specifically, we apply a Discrete Fourier transform (DFT) using a Fast Fourier Transform algorithm (FFT) [29] on the undulations, since they present a sinusoidal form (see Figure 5).

DFT was performed with the Python library NumPy [30]. The python algorithm that performs DFT requires a one-dimensional signal (input) over a specified domain. We chose the domain to be the length of the column, and our signal is the wave undulation. We specified an arbitrary number (10) of sine frequency components that may comprise our signal, and we determined which of these components contributes the most (leading frequency component). Once the leading frequency component was found, an inverse Fourier computation allowed us to transform the leading frequency component back to a one-dimensional signal. In summary, we disassembled the signal into multiple contributing signals of various frequencies then kept the largest contributor for our purposes. This (largest) isolated mode is the leading factor of deformation on the flange.

We can extract the nodal displacements in two ways. For this study, we extracted them from Abaqus simulation. However, in an experimental or field implementation setting, we would extract them from a laser deflectometer or similar instrumentation that can be used to monitor the flange displacements. In any case, extraction of nodal displacements is always followed by a DFT to isolate the leading modes.

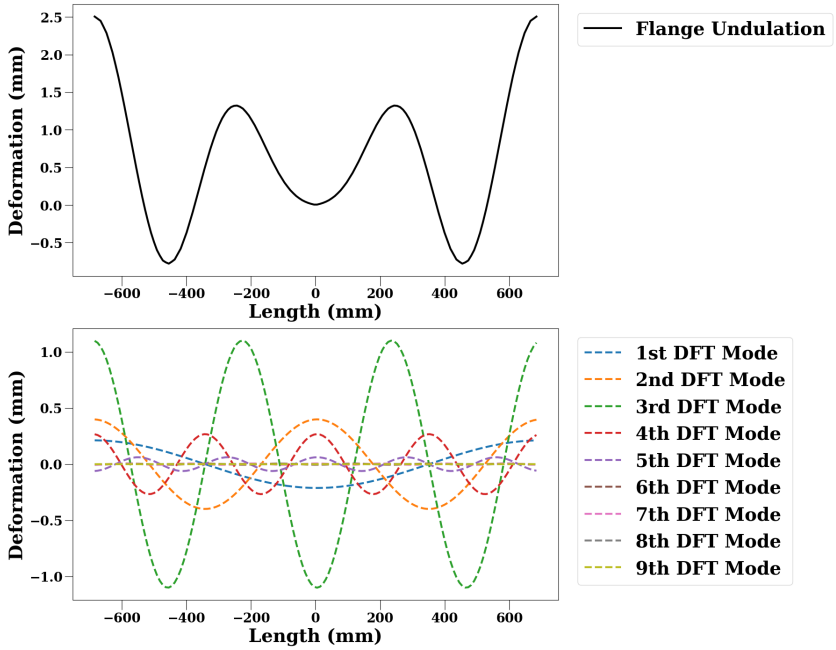

Figure 5. Flange deformation along the length of a column with slenderness $\lambda = 0.6$ (top chart). Discrete Fourier transform modes 1–9 of flange deformation when the local and Euler imperfections are 0.1 and 0.5, respectively (bottom chart). For the specific case shown here, the leading frequency corresponds to the 3rd DFT mode (with the largest amplitude), which was used in the ML algorithm.

In this work, Abaqus nodal displacements at 30% of service load were used to create the undulations. DFT and subsequent inverse DFT were used to extract the leading sinusoidal form of the nodal displacement. The displacement of each node (150 in total) was used as an input feature in our ML pipeline, and the length of the column led to a total of 151 experimentally observable features that were used to train our ML model to recognize IS columns.

In total, our deep neural network comprises 2750 columns, which have 151 features to describe whether they are IS or not. All features are used as an initial input and are fed forward to our four hidden units of decreasing nodes for a total of 100 epochs to maximize accuracy and efficiency. The output must be cast as a binary classification of 0 and 1, where 0 indicates a NIS column and 1 indicates an IS column. The model parameters at the final epoch are used to evaluate the test data (1125 columns) which has the same number of input features and is not used for training but set aside for testing the accuracy of the trained NN. To reduce the model complexity and the overall number of training parameters, it may be possible to use weight quantization or network pruning techniques [31].

3. Results

The results of an application of any ML algorithm to a dataset is typically presented via metrics that describe how well the trained model captures the necessary behavior the user expects to observe. One of the most common evaluation metrics is accuracy, which measures the overall correctness of a model's prediction. Accuracy is the ratio of correctly predicted instances versus the total number of instances. When both training and test sets achieve high accuracy, the results indicate a well-trained ML algorithm, which is expected to generalize well on similar data. Otherwise, when there is a large difference between training accuracy and testing accuracy, the algorithm may suffer from over- or under-training.

In our case, accuracy is shown in Figure 6 for both the training set and the validation set. Both datasets follow the same pattern and reach an agreement with 95% training accuracy. The training set reaches high accuracy (about 90%) in a few epochs and continues to improve with additional training. Since the weights are well-trained, when the validation

set is passed through the ML algorithm, the validation accuracy immediately reaches values similar to the training accuracy.

A second evaluation metric that is used for assessment of ML applications is the response of the loss function. Generally, the loss function should be minimized in each epoch, and it is desirable to achieve similar scores in both training and validation sets. The combination of accuracy and loss evaluation metrics can indicate how well an algorithm will generalize when new data are introduced.

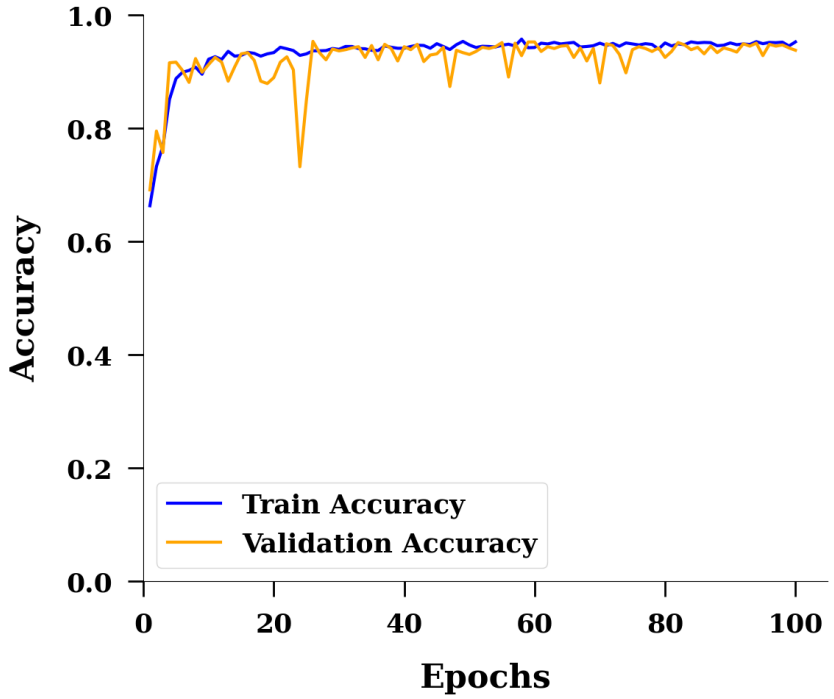

Figure 6. Imperfection sensitivity accuracy of the proposed ML implementation. Accuracy and epochs are both dimensionless.

The loss score for our training and validation phases can be seen in Figure 7. Similar to the accuracy graph (Figure 6), the loss of training and validation sets follows the same trend. The training set slowly plateaus around 10%, with the validation set achieving almost the same value. Because of the matching values, the algorithm can be considered to be well-trained.

The variations in the validation dataset loss are generally indicative of overfitting in the training set, i.e., the algorithm would not generalize well in unknown test data. Validation datasets are also used to fine-tune the training model by making sure the errors are minimized during each epoch. There are numerous ways to reduce the variations, which include L2 regularization, early stopping implementation, batch normalization, or adjustment of the learning rate with a more dynamic implementation. Such results can be found in the supplementary material. However, the accuracy value is in agreement with training, which indicates avoidance of overfitting.

An in-depth look at the accuracy score is provided in Figures 8 and 9, showing the *confusion matrix* of the training and validation datasets, respectively. A confusion matrix is a matrix that summarizes the performance of an ML algorithm. It displays the percentage of accurate (true positives and true negatives) and inaccurate (false positives and false negatives) instances in the dataset.

In Figure 8, the accuracy of the training set is shown, measured as the percentage of accurate predicted labels vs. true labels in our ML pipeline. The overall accuracy is 95.32% in the training set, with 6% of imperfection-sensitive columns being misclassified. The top-left corner shows a 96.1% accuracy in detection of NIS columns (NOT I.S.-NOT I.S. in

the axes). The bottom-right corner shows a 93.6% accuracy in detecting IS columns (I.S.-I.S. in the axes).

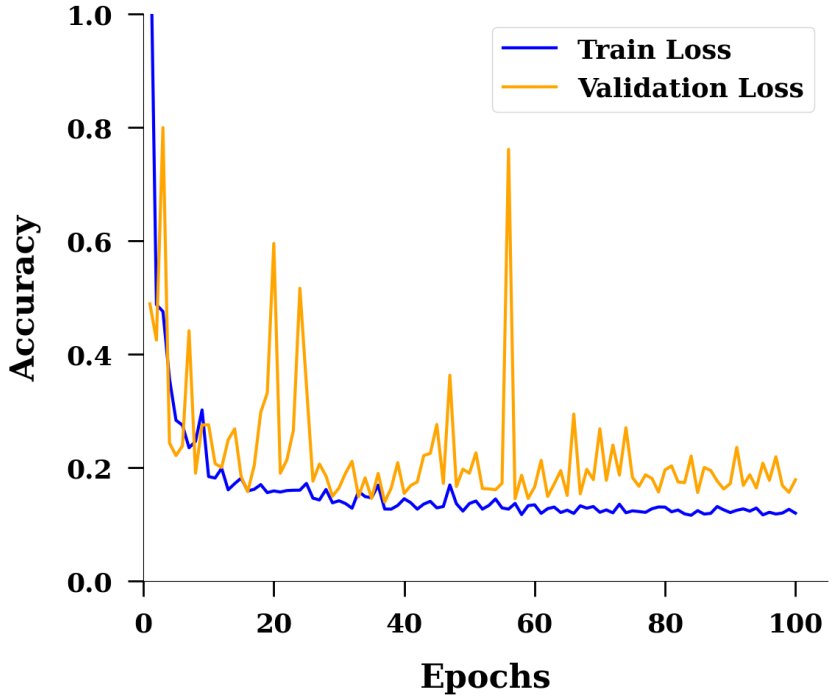

Figure 7. Imperfection sensitivity loss of the proposed ML implementation. Accuracy and epochs are both dimensionless.

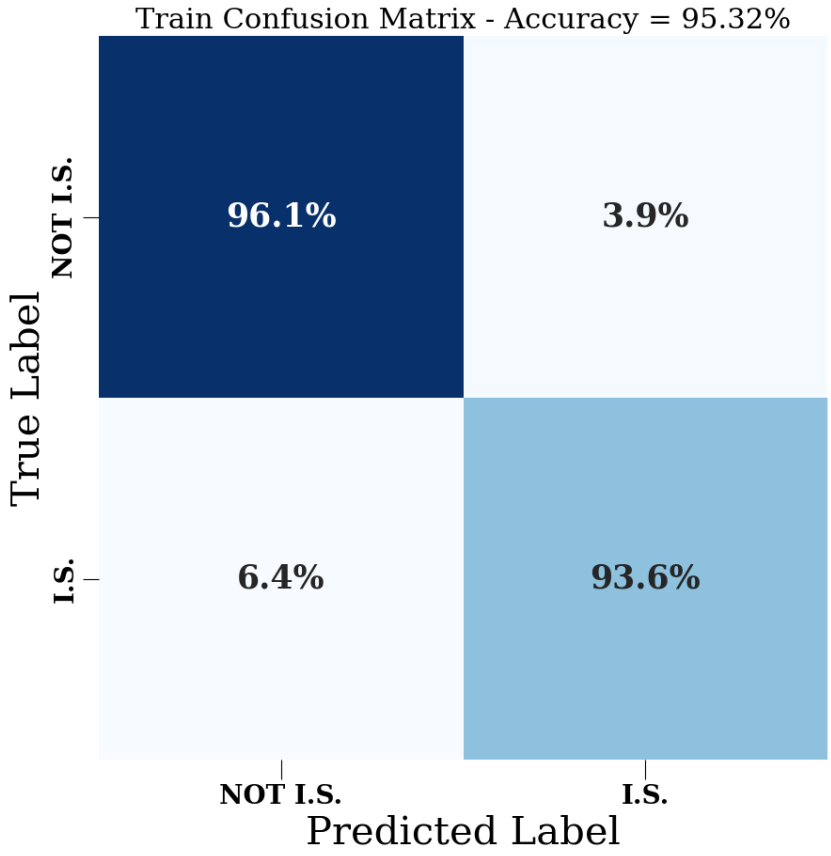

Figure 8. Training confusion matrix for prediction of imperfection sensitivity.

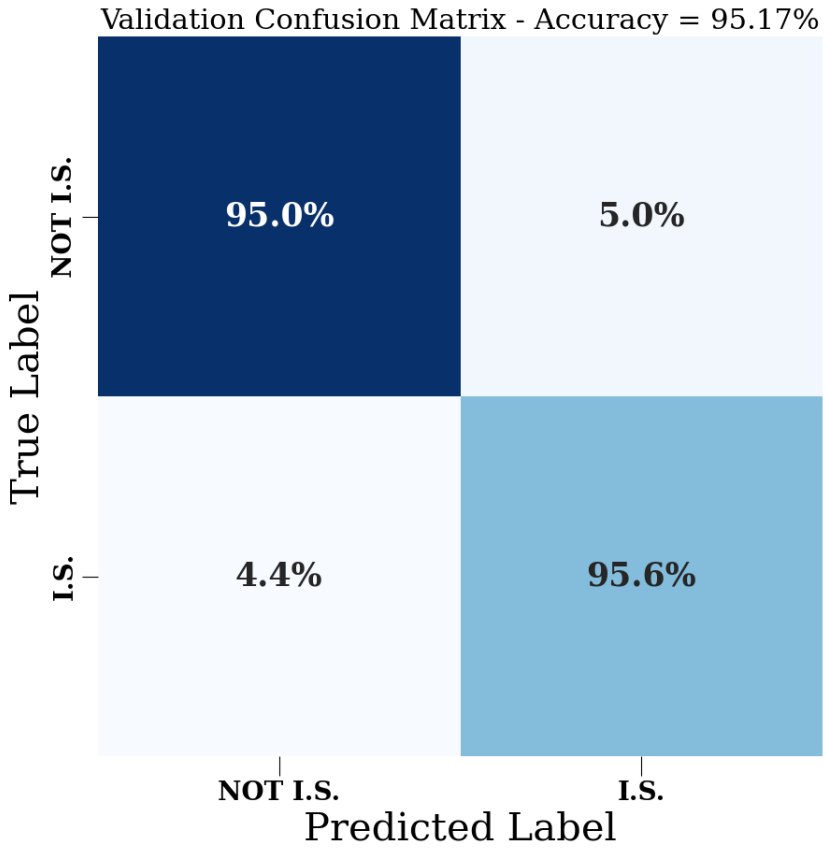

Figure 9. Validation confusion matrix for prediction of imperfection sensitivity.

Similarly to the training dataset, the validation confusion matrix (Figure 9) shows that the validation accuracy is 95.17%. This is encouraging based on the modest number of samples that were used in this work for training and testing. Generally, the amount of training data required for an ML algorithm to achieve high accuracy is very large. For a complex problem such as imperfection sensitivity detection, accuracy scores that exceed 95% in both training and in validation suggest a strong likelihood of applicability of the proposed ML model for field-testing and further development of the current project. This shows the validity of the algorithm for this study and that it could likely be generalized for similar applications.

Based on the high accuracy in training and validation datasets we observed, it can be assumed that detection of imperfection-sensitive columns with inexpensive instrumentation is possible and that it can be potentially extended to real-time applications provided the necessary measurements can be provided. Once a column is identified as IS, it is almost certain (95% certain) that the column is at risk, which should trigger a warning and/or corrective measures.

All of the above results assume that training contains 70% of the dataset, and the rest (30%) is used for validation as a form of early testing. A further step is to split the validation dataset into 20% for validation and 10% for testing. This split aims to further enhance our conclusions. The overall accuracy in training, validation, and testing datasets when we used the 70–20–10 split described above, is shown in Table 4. The difference in values can be explained from the required shuffling that happens when the datasets are split, as well as the overall decreased amount of data that is used to validate the training set. The accuracy value differences are negligible and can be further enhanced with various optimization techniques that are not used in our model. The agreement between training, validation, and testing datasets shows that there was no overfitting.

Table 4. Training, validation, and testing dataset accuracy.

Accuracy	Training	Validation	Testing
%	94.60	94.86	93.71

Further models were briefly explored with the accuracies of training and validation summarized in Table 5. The models were chosen for their applicability in binary classification problems. We see that Logistic Regression [32] and Support Vector Machines (SVMs) [33] have very low accuracy compared to our MLP implementation. A Random Forest implementation [34] shows exemplary accuracy in training, but there is a difference in validation accuracy (5%). In terms of the training time needed for each implementation, it is expected that neural networks will take the most time due to the complexity of the model. The rest of the models show extremely fast solutions (less than a second). Therefore, the only model that is arguably better than our MLP implementation is the Random Forest implementation.

Table 5. Training and validation accuracy for various ML models.

	MLP	Logistic Regression	Random Forest	SVM
Train. Accuracy	95.32%	70.07%	100%	75.15%
Val. Accuracy	95.17%	69.04%	94.33%	75.13%
Time	27.99 s	0.093 s	0.66 s	0.27 s

4. Discussion

Thin-walled structures may become imperfection-sensitive during their service life due to a variety of events such as aging, unanticipated damage, and fatigue. Detection of strength reduction is an extremely important aspect in SHM. Further, a non-destructive and inexpensive method of recognizing such reduction is equally important. We explored the applicability of ML algorithms in SHM with the goal of identifying imperfection-sensitive columns that are prone to experiencing a significant reduction in strength. The ML algorithm is trained with synthetic data. Then, it is proposed that it be used along with simple and inexpensive SHM data collection from in-service columns. As reported in Table 5, ML is able to provide fast prediction of imperfection sensitivity, which can be used to provide an instantaneous alert about the imperfection sensitivity of the in-service column. Generalization to other buckling-prone structures, such as shells, should be immediate.

The scope of this study was focused on the prediction of imperfection sensitivity in fiber-reinforced composite columns of various lengths. Localized damage and different material properties were beyond the scope of this work; while our scope was limited, the cases considered cover a wide spectrum of internal and external imperfections that may be found in an experimental setting.

Theoretical solutions for loaded columns reveal two possible states: Euler mode and local mode. For slender columns, an isolated Euler mode exists with overall lateral deflection, and there is an analytical solution that predicts the maximum load based on the column geometry, length, and material properties. For stubby columns, the flanges of the column deform to an isolated local mode, and the maximum load is predicted to be independent of column length. Experimental observations have shown that the theoretical solution does not hold true for the intermediate column length where the Euler and local modes combine to produce a significant reduction in strength.

Local and Euler modes depend on the critical length of the column. If $L < Lcr$, then the theoretical solution dictates that $P_{max} = P_{cr}$. If $L > L_{cr}$, then the maximum load follows the Euler solution. Experimental findings [9] have shown that this is not the case when L/Lcr is close to 1.0. Instead, the column experiences the interaction of local and Euler modes with simultaneous lateral and flange deformations. For such columns, the experimentally reported failure load is significantly less than the theoretical load. This is a serious issue,

since the maximum load a column can withstand is significantly reduced when compared to the service load.

Aging structures that have accumulated internal and external imperfections may become imperfection-sensitive and thus experience a significant reduction in strength with respect to that of the perfect column. Current research on SHM is focused on real-time identification [35–37]. However, the proposed instrumentation is either expensive, or the data collection and data transmission requirements are burdensome for real-time applications.

In this work, we propose a laser deflectometer, or similar, to monitor a few deflections, as few as four deflections. Fourier Transform was used to capture the leading modes. Then, an inverse DFT was used on the leading frequencies of deformation. To apply the ML algorithm, the length of the column is also needed, but the length is always readily available. With these easily captured features, the proposed ML algorithm is able to accurately predict whether, in service, the column has become imperfection-sensitive or not. Imperfection-sensitive columns exhibit behaviors such that, due to internal or external factors, the as-damaged failure load does not match the expected failure load of the perfect column.

In the proposed methodology, as long as a damaged column produces deformations, the algorithm can reliably predict with 95% accuracy whether imperfection sensitivity is present or not. The nature of the damage will remain unknown until field-testing and inspection of the column takes place. Therefore, the objective of this research is the development of a tool that can reliably, inexpensively, and in real time provide a warning for further inspection.

In Figures 6–9, we observe the overall efficiency of the ML algorithm. Specifically, training on synthetic data for 2625 columns, we achieved an overall accuracy of 95%, which also extends to the testing data. The loss function, as shown in Figure 7, further validates the results. Similar values of accuracy and loss function in both training and testing datasets indicate a well-trained ML algorithm which can be used on data that have similar features. The confusion matrices shown in Figures 8 and 9 further expand and support the accuracy evaluation metric for identifying imperfection-sensitive columns.

Moreover, the incorrect prediction of imperfection-sensitive columns (false positives, 3.9%) is a good indicator. This means that only 3.9% of columns were reported as damaged when they actually were not. This is a small incurred cost compared to the overall structural safety enhancement provided by 95% accuracy prediction of at-risk columns. Of course, accuracy can always be further improved with additional training data points.

Comparison with other ML methods (Table 5) shows that only MLP and Random Forests (RFs) can reliably predict imperfection sensitivity, and they can be trained faster. However, it is known that the RF method has limited scalability for more complex systems. It is also possible that training of the RF method was easier because our features were specific, and our training set contained a large amount of data. For the case with only four deflections being field-measured, the RF method may have lower accuracy than MLP because, when compared to neural networks, the RF method does not consider complex connections.

As further work, we propose comparing the findings of this study with lab experiments. For example, Abdeljaber et al. [38] collected vibration data from loosened bolts in a frame and used such data to predict which bolts were actually loosened. Experiments in a controlled setting can provide a more concrete view of the applicability of our proposed method, prior to on-field testing.

Author Contributions: Conceptualization, E.J.B.; Methodology, E.J.B.; Software, M.T.; Validation, M.T.; Writing – original draft, M.T.; Writing – review & editing, E.J.B.; Supervision, E.J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available in a publicly accessible repository at https://github.com/mtzimas92/Imperfection-Sensitivity-Detection/tree/main (accessed on 20 March 2024).

Acknowledgments: The second author gratefully acknowledges the support of Universidad Carlos III de Madrid (UC3M) in the framework of the "cátedras de excelencia" program and further wishes to dedicate this work to the memory of the late José Fernández-Sáez, UC3M.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript, in order of appearance in the text:

ML	Machine learning
SHM	Structural health monitoring
FEA	Finite Element Analysis
FRP	Fiber-reinforced plastic
WF	Wide flange
IS	Imperfection-sensitive
NIS	Non-imperfection-sensitive
NGA	Non-linear geometric analysis
AI	Artificial Intelligence
NN	Neural network
MLP	Multilayer perceptron
RP	Reference point
DFT	Discrete Fourier transform
FFT	Fast Fourier Transform
SVM	Support Vector Machine
Symbols	
λ	Slenderness of a column, ratio of column length over critical length
P_L	Local buckling load
P_E	Euler buckling load
L	Column length
L_{cr}	Column critical length
E	Young's modulus
I	Second moment of inertia for the cross-section
K	Parameter that changes based on end-supports of a column
P_{cr}	Column critical load
N	Total number of samples
P_{max}	Column peak load
P_{final}	Column final load
u_{final}	Column final displacement
u_{max}	Column displacement at peak load

References

1. Bonopera, M.; Chang, K.C.; Chen, C.C.; Lin, T.K.; Tullini, N. Compressive column load identification in steel space frames using second-order deflection-based methods. *Int. J. Struct. Stab. Dyn.* **2018**, *18*, 1850092. [CrossRef]
2. Dassault Systèmes. Abaqus 2020 Documentation. 2020. Available online: https://www.3ds.com/ (accessed on 1 April 2024).
3. Barbero, E.J. Buckling Mode Interaction in Pultruded Composite Columns. YouTube. 2019. Available online: https://youtu.be/Nl8YRFQMcfg (accessed on 1 April 2024).
4. Eidukynas, D.; Adumitroaie, A.; Griškevičius, P.; Grigaliunas, V.; Vaitkūnas, T. Finite Element Model Updating Approach for Structural Health Monitoring of Lightweight Structures Using Response Surface Optimization. In *Proceedings of the IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2022; Volume 1239, p. 012002.
5. Budiansky, B. Theory of buckling and post-buckling behavior of elastic structures. *Adv. Appl. Mech.* **1974**, *14*, 1–65.
6. Barbero, E.; Tomblin, J. A phenomenological design equation for FRP columns with interaction between local and global buckling. *Thin-Walled Struct.* **1994**, *18*, 117–131. [CrossRef]
7. Ascione, F.; Feo, L.; Lamberti, M.; Minghini, F.; Tullini, N. A closed-form equation for the local buckling moment of pultruded FRP I-beams in major-axis bending. *Compos. Part B Eng.* **2016**, *97*, 292–299. [CrossRef]
8. Dos Santos, R.R.; Castro, S.G. Lightweight design of variable-stiffness cylinders with reduced imperfection sensitivity enabled by continuous tow shearing and machine learning. *Materials* **2022**, *15*, 4117. [CrossRef] [PubMed]
9. Barbero, E.J. Prediction of buckling-mode interaction in composite columns. *Mech. Compos. Mater. Struct.* **2000**, *7*, 269–284. [CrossRef]

10. Sonti, S.S.; Barbero, E.J. Material characterization of pultruded laminates and shapes. *J. Reinf. Plast. Compos.* **1996**, *15*, 701–717. [CrossRef]
11. Barbero, E.J. *Finite Element Analysis of Composite Materials Using Abaqus*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2023.
12. Barbero, E.; Sonti, S. Micromechanical models for pultruded composite beams. In Proceedings of the 32nd Structures, Structural Dynamics, and Materials Conference, Baltimore, MD, USA, 8–10 April 1991; p. 1045.
13. Vasios, N. Nonlinear Analysis of Structures. The Arc Length Method: Formulation, Implementation and Applications/Nikolaos Vasios. Available online: https://scholar.harvard.edu/sites/scholar.harvard.edu/files/vasios/files/ArcLength.pdf (accessed on 1 April 2024).
14. Barbero, E.J.; Raftoyiannis, I.G. Local buckling of FRP beams and columns. *J. Mater. Civ. Eng.* **1993**, *5*, 339–355. [CrossRef]
15. Alpaydin, E. *Machine Learning*; MIT Press: Cambridge, MA, USA, 2021.
16. Pak, M.; Kim, S. A review of deep learning in image recognition. In Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), Kuta Bali, Indonesia, 8–10 August 2017; pp. 1–3.
17. Papanikolaou, S.; Tzimas, M.; Reid, A.C.; Langer, S.A. Spatial strain correlations, machine learning, and deformation history in crystal plasticity. *Phys. Rev. E* **2019**, *99*, 053003. [CrossRef]
18. Papanikolaou, S.; Tzimas, M. Effects of rate, size, and prior deformation in microcrystal plasticity. In *Mechanics and Physics of Solids at Micro-and Nano-Scales*; Wiley Online Library: Mew York, NY, USA, 2019; pp. 25–54.
19. Megalooikonomou, K.G.; Beligiannis, G.N. Random Forests Machine Learning Applied to PEER Structural Performance Experimental Columns Database. *Appl. Sci.* **2023**, *13*, 12821. [CrossRef]
20. Tran, V.L.; Lee, T.H.; Nguyen, D.D.; Nguyen, T.H.; Vu, Q.V.; Phan, H.T. Failure Mode Identification and Shear Strength Prediction of Rectangular Hollow RC Columns Using Novel Hybrid Machine Learning Models. *Buildings* **2023**, *13*, 2914. [CrossRef]
21. Phan, V.T.; Tran, V.L.; Nguyen, V.Q.; Nguyen, D.D. Machine learning models for predicting shear strength and identifying failure modes of rectangular RC columns. *Buildings* **2022**, *12*, 1493. [CrossRef]
22. Cakiroglu, C.; Islam, K.; Bekdaş, G.; Kim, S.; Geem, Z.W. Interpretable machine learning algorithms to predict the axial capacity of FRP-reinforced concrete columns. *Materials* **2022**, *15*, 2742. [CrossRef] [PubMed]
23. Alpaydin, E. *Introduction to Machine Learning, Ed.*; Massachusetts Institutes of Technology: Cambridge, MA, USA, 2010.
24. Anderson, J.A. *An Introduction to Neural Networks*; MIT Press: Cambridge, MA, USA, 1995.
25. Cichy, R.M.; Kaiser, D. Deep neural networks as scientific models. *Trends Cogn. Sci.* **2019**, *23*, 305–317. [CrossRef] [PubMed]
26. Janocha, K.; Czarnecki, W.M. On loss functions for deep neural networks in classification. *arXiv* **2017**, arXiv:1702.05659.
27. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://www.tensorflow.org/ (accessed on 1 April 2024).
28. Sharma, S.; Sharma, S.; Athaiya, A. Activation functions in neural networks. *Towards Data Sci.* **2017**, *6*, 310–316. [CrossRef]
29. Nussbaumer, H.J.; Nussbaumer, H.J. *The Fast Fourier Transform*; Springer: New York, NY, USA, 1982.
30. Harris, C.R.; Millman, K.J.; Van Der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef] [PubMed]
31. Tang, Z.; Luo, L.; Xie, B.; Zhu, Y.; Zhao, R.; Bi, L.; Lu, C. Automatic sparse connectivity learning for neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 7350–7364. [CrossRef] [PubMed]
32. Kleinbaum, D.G.; Dietz, K.; Gail, M.; Klein, M.; Klein, M. In *Logistic Regression*; Springer: New York, NY, USA, 2002.
33. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support vector machines. *IEEE Intell. Syst. Their Appl.* **1998**, *13*, 18–28. [CrossRef]
34. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
35. Mishra, M.; Lourenço, P.B.; Ramana, G.V. Structural health monitoring of civil engineering structures by using the internet of things: A review. *J. Build. Eng.* **2022**, *48*, 103954. [CrossRef]
36. Flah, M.; Nunez, I.; Ben Chaabene, W.; Nehdi, M.L. Machine learning algorithms in civil structural health monitoring: A systematic review. *Arch. Comput. Methods Eng.* **2021**, *28*, 2621–2643. [CrossRef]
37. Tibaduiza, D.; Torres-Arredondo, M.Á.; Vitola, J.; Anaya, M.; Pozo, F. A damage classification approach for structural health monitoring using machine learning. *Complexity* **2018**, *2018*, 1–14. [CrossRef]
38. Abdeljaber, O.; Avci, O.; Kiranyaz, S.; Gabbouj, M.; Inman, D.J. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *J. Sound Vib.* **2017**, *388*, 154–170. [CrossRef]