SAND2018-6582C

Efficient Random Vibration Analysis of Nonlinear Systems with Long Short-Term Memory Networks for Uncertainty Quantification

D. A. Najera-Flores[1], A. R. Brink[2]
[1] ATA Engineering, Inc.
San Diego, CA, USA
e-mail: **dnajera@ata-e.com**
[2] Sandia National Laboratories
Albuquerque, NM, USA

Abstract

Complex mechanical structures are often subjected to random vibration environments. One strategy to analyze these nonlinear structures numerically is to use finite element analysis with an explicit solver to resolve interactions in the time domain. However, this approach is impractical because the solver is conditionally stable and requires thousands of iterations to resolve the contact algorithms. As a result, only short runs can be performed practically because of the extremely long runtime needed to obtain sufficient sampling for long-time statistics. The proposed approach uses a machine learning algorithm known as the Long Short-Term Memory (LSTM) network to model the response of the nonlinear system to random input. The LSTM extends the capability of the explicit solver approach by taking short samples and extending them to arbitrarily long signals. The efficient LSTM algorithm enables the capability to perform Monte Carlo simulations to quantify model-form and aleatoric uncertainty due to the random input.

1 Introduction

Random vibration is an important load environment to consider during the design of mechanical, aerospace and civil structures as it can lead to fatigue type failures. As the name implies, random vibration is a stochastic load that is best described in the frequency domain using a power spectral density function (PSD). For linear systems, the PSD of the response is calculated as

$$[S_X(\omega)] = [H(\omega)][S_F(\omega)][H^*(\omega)]^T, \tag{1}$$

where $S_X(\omega)$ is the response PSD, $H(\omega)$ is the frequency response function (FRF), $S_F(\omega)$ is the input PSD, and $[\cdot]^*$ is the complex conjugate [1]. An exhaustive discussion on random vibration of linear systems is found in [1] and [2]. In practice, it is rare to find a structure that behaves in a truly linear way. Often the nonlinearities are weak and do not contribute to the overall response, thus the linear assumption holds. However, there are applications and structures in which the nonlinearities play an important role, such as geometric nonlinearity in panel flutter [3], hysteretic nonlinearity in vehicle suspension [4], and nonlinear stiffness and damping from frictional joints [5]. Joint probability distribution functions (PDF) inclusive of the time varying nature of nonlinear systems can be numerically simulated using several existing techniques, such as perturbation techniques [6], equivalent linearization [7], multi-Gaussian closures [8], stochastic averaging [9], or seeking the solution to the Fokker-Planck equation for Markov vector processes [10], [11]. These techniques are only valid for narrow classes of vibration input, such as stationary Gaussian, and can only be reasonably solved for systems with limited degrees of freedom (DOF). The work presented in this paper is motivated by the need to develop an efficient method to calculate the structural response of nonlinear mechanical systems with a large number of DOF subjected to any class of random vibration excitations. Currently for large numerical models of complex structures, explicit time integration is performed using finite element analysis (FEA). A major drawback of this method is that explicit solvers are

conditionally stable because their smallest allowable time step is driven by the smallest characteristic element size of the finite element model (FEM). Since FEMs that include contact nonlinearities require high mesh refinement to accurately resolve conditions at interfaces, the stable timestep becomes small enough to make analysis of long duration signals intractable. As a result, random analysis is typically performed using a short realization of the random environment. This approach is problematic because the time signal statistics may not be captured by a short realization. Moreover, a single realization may not be representative of the worst case for a nonlinear structure where there is path dependency. These challenges motivate the need for a more efficient approach for obtaining structural response of nonlinear structures to random vibration inputs.

The technique presented in this paper aims to provide a novel and efficient methodology to conduct numerical analysis of nonlinear structures subjected to random inputs. The method consists of using machine learning techniques to accomplish the objective. In particular, the Long Short-Term Memory (LSTM) network is used to extend the structural response of a short duration random signal to a much longer time signal from which significant statistical features can be extracted. The remainder of this paper will introduce the mechanics of a LSTM neural network, then apply them to a system with a strong and weak contact nonlinearity.

2 Methodology

2.1 Long Short-Term Memory Networks

The field of deep learning has been applied with success to a broad set of applications in recent years [12]. The basis for deep learning is algorithms that learn from experience when exposed to data. A commonly used tool in deep learning is a deep neural network which is built by stacking sequential layers of parallel operations, referred to as units. The sequential traversal through the algorithm allows the program to refer to earlier operations. These units act as storage for state information and provide a structure to organize the network's processes. As such, deep neural networks learn both hierarchical representations of data, and the structural organization of the computer program that process the data.

A special case of deep neural networks is the recurrent neural network (RNN). RNNs are a family of neural networks specialized in processing sequential data [13]. The main advantage of an RNN over a traditional neural network is its ability to share model parameters across different timesteps. This capability allows the network to learn relations between timesteps. A more specialized type of RNN is the long short-term memory (LSTM) network. The LSTM, which is a type of gated RNN, was originally proposed by Hochreiter and Schmidhuber [14], and alleviates many of the computational challenges of the traditional RNN [15]. Using gates to control the flow of information through time, the LSTM has self-loops that allow it to "forget" or "remember" information about the past. Additionally, by using bidirectionality, the LSTM can also learn how the "future" affects the current state. LSTM networks have been successful in many applications, including speech recognition [16], handwriting recognition and generation [17], machine translation [18], image captioning [19], among others.

A diagram of a single LSTM cell can be seen in Figure 1.

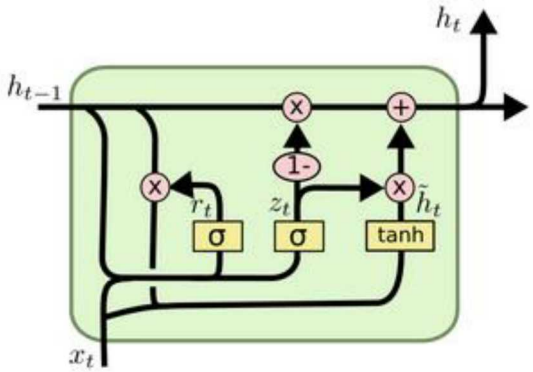

Figure 1. A single LSTM cell *[20]*.

The forget gate unit $f_i^{(t)}$(for timestep *t* and cell *i*) is computed as

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right), \tag{2}$$

where b_i^f, U_i^f, W_i^f are the biases, input weights and recurrent weights for the forget states, respectively. The input vector, represented by $x^{(t)}$ and $h^{(t)}$ (e.g., the input and output acceleration of a dynamic process), is the current hidden layer vector. The $\sigma()$ operator is the sigmoid function. The LSTM cell internal state is updated by

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^t \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right), \tag{3}$$

where b_i , U_i , W_i are the biases, input weights, and recurrent weights into the LSTM network. The nonlinear sigmoid operation is weighted by the external input gate. The external input gate is computed similarly to the forget gate but with its own set of parameters, such that

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right). \tag{4}$$

The output $h_i^{(t)}$ of the LSTM cell can be controller via the output gate $q_i^{(t)}$

$$h_i^{(t)} = \tanh(s_i^{(t)})\, q_i^{(t)}, \tag{5}$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right). \tag{6}$$

In the above equations, all parameters are learned through a process known as backpropagation through time (BPTT) [13]. BPTT is the process by which gradients are computed through the nodes of the computational graph with respect to a loss function. The loss function is typically the negative log likelihood function, where the current timestep is predicted from past and current inputs, and past outputs, such as

$$L^t = -logP\left(y^{(t)} | x^{(1)}, \ldots, x^{(t)}, y^{(1)}, \ldots, y^{(t-1)}\right). \tag{7}$$

2.2 Dropout and Model-Form Uncertainty

Dropout is a method of regularization typically used to reduce the model's generalization error [21]. It is usually implemented by masking certain hidden units chosen at random, based on a uniform probability defined by the user. The masked hidden unit is dropped, reducing the model's effective capacity. At each epoch, different units are dropped, which forces the model to be more resilient; each hidden unit must be able to perform well in the absence of other units. The implication is that the hidden units are optimized to perform well in many contexts, resulting in a more generalizable network [22].

Gal and Ghahramani [23] showed that dropout can be re-interpreted as performing approximate Bayesian inference in deep Gaussian processes. They developed the theoretical framework to show that performing backpropagation through a neural network is analogous to minimizing the Kullback-Leilbler divergence between an approximate distribution and the posterior of a deep Gaussian process. Therefore, a trained deep network with dropout included essentially functions as a deep Gaussian process. The implication is that activating dropout during analysis provides a stochastic prediction. A family of predictions can be obtained with a Monte Carlo simulation by performing numerous forward passes through the stochastic network. The resulting statistical distribution of predictions is an expression of the model-form error.

3 Case Studies

3.1 Intermittent Contact Problem

The contact problem under consideration in this section is a three degree of freedom (DOF) system, as illustrated in Figure 2. It consists of three differing masses connected by springs. A penalty spring was used to model contact between M_2 and M_3, with a small initial gap. An input random vibration acceleration was applied at one end while the other end remained fixed. Response was recovered at M_1. This location was chosen instead of M_2 to avoid contaminating the data with artificial peaks in the response due to the arbitrary penalty stiffness. The response of M_1 was still affected by the contact interaction but without the numerical artifacts of the contact algorithm. The masses and springs were set to arbitrary values such that there was significant contact throughout the excitation. The masses M_1, M_2, and M_3 were set to 1, 10, and 2, respectively. The spring stiffnesses K_1, K_2, and K_3 were set to 100, 2000, and 100, respectively.

The goal of this notional problem was to exercise nonlinear response due to intermittent contact interactions. The expectation is that the contact problem is harder to predict due to the non-smooth response associated with impact between the two masses. The objective is to predict the acceleration response at M_1, given the input random vibration. The three DOF system was implemented and solved in Matlab using ODE45.

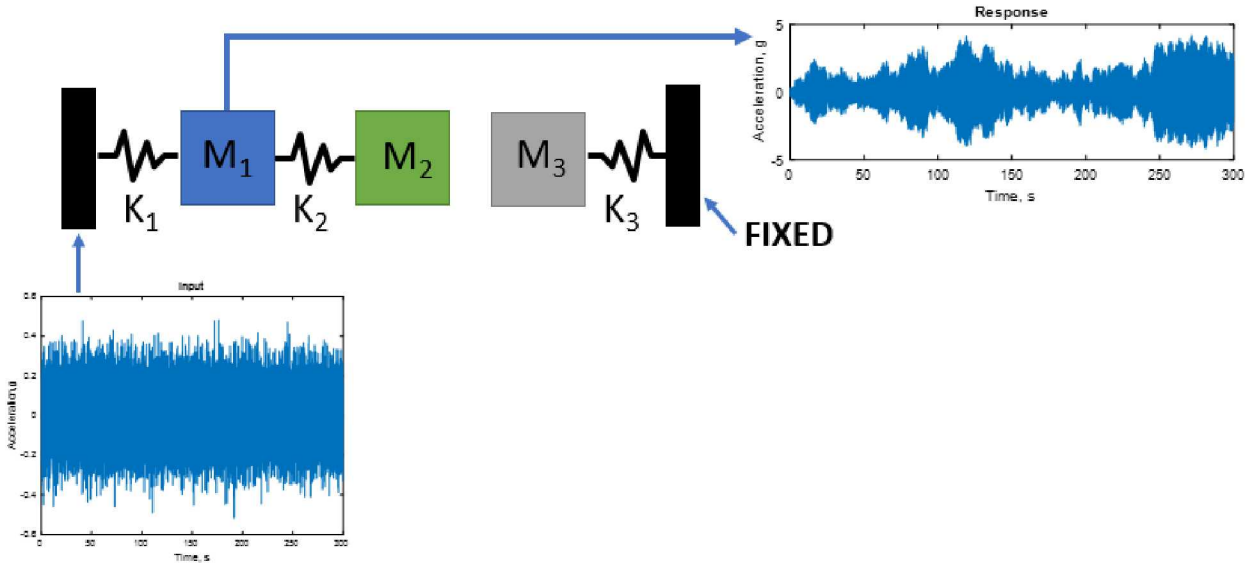

Figure 2. Diagram of the three DOF contact problem with boundary conditions indicated.

3.1.1 LSTM Architecture

The LSTM architecture used to model the contact problem was a deep bidirectional network with dropout layers, and a time distributed dense output layer. The data (input and response) was divided in *m* segments that were 100 points long (each point corresponding to a timestep). The LSTM was trained by using these 100-step segments. The input to the LSTM was a 2x100 vector containing the segment corresponding to the i^{th} step excitation and the segment corresponding to the response of the $(i\text{-}1)^{th}$ segment. The output to the LSTM was a 1x100 vector containing the segment corresponding to the i^{th} step response. Figure 3 shows an example of this process; the LSTM network takes in the excitation at the segment we are trying to predict and the response of the previous segment (blue) and outputs the response at the current segment (orange).

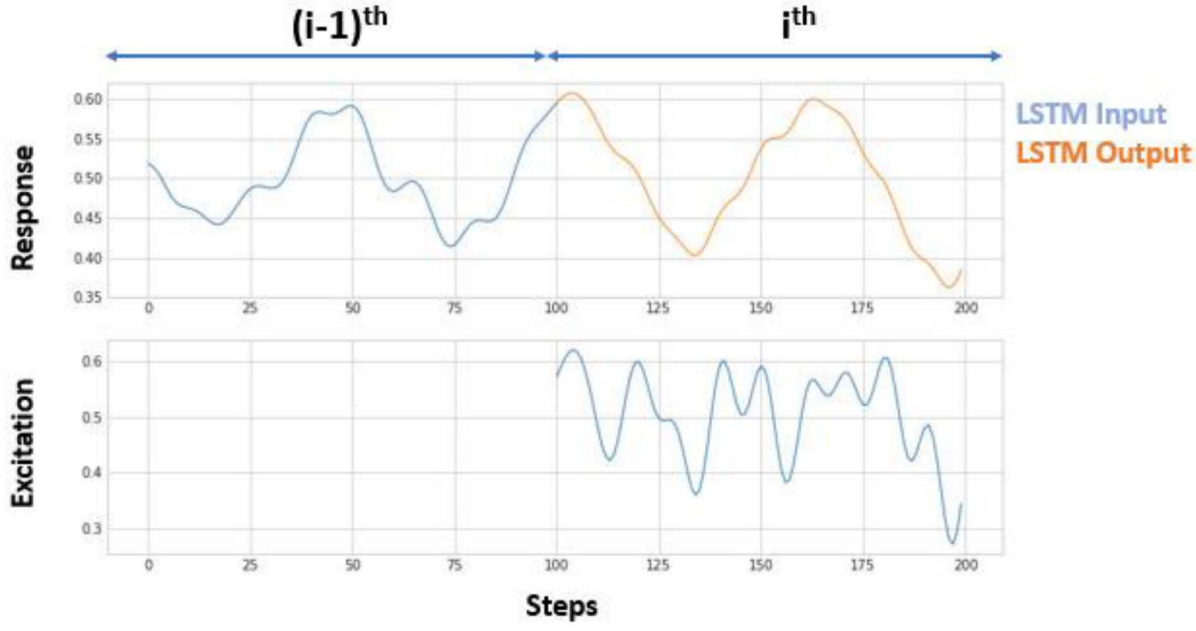

Figure 3. Typical input/output data used for training the LSTM network.

Figure 4 shows a diagram representing the LSTM architecture used. The x variables represent the input time history, while the y variables represent the output time history. The diagram shows the prediction of a single segment. In this case, the T superscript would be 100, representing the number of points in a segment. Each row of the diagram represents a layer. The general architecture used contained layers of bidirectional LSTMs, followed by forward LSTMs, followed by a time distributed dense output layer. The data was scaled to be between 0 and 1, so that rectified linear unit (ReLu) activation functions could be used throughout the network. After each layer, a dropout layer was added for regularization. As seen in the diagram, output of the LSTM units is shared across the entire network.

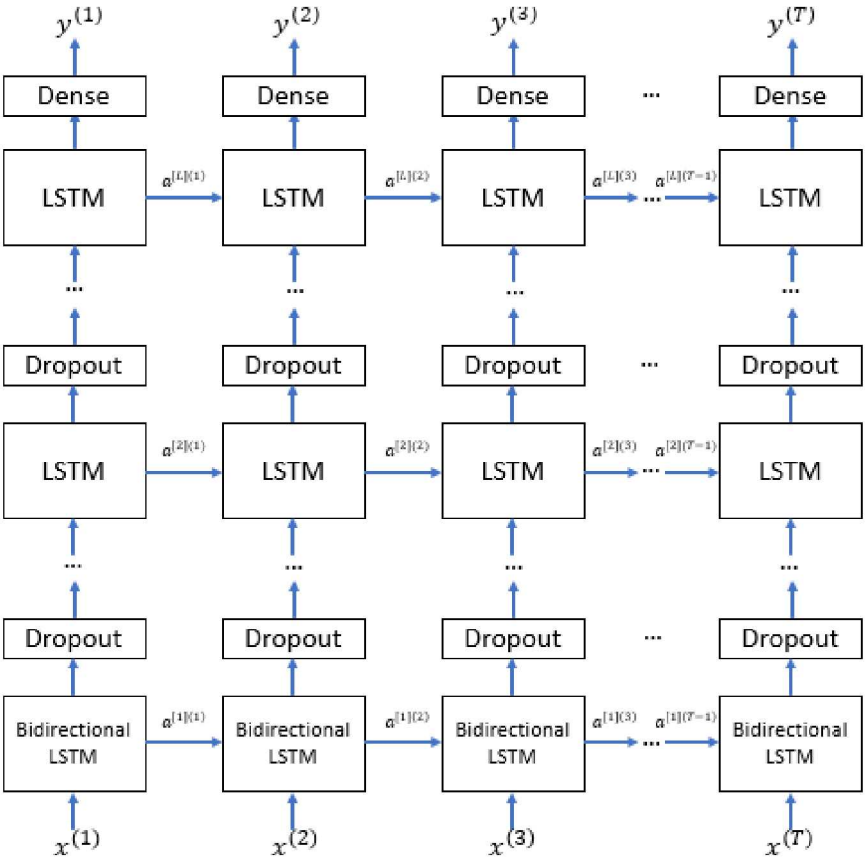

Figure 4. Schematic of the LSTM architecture used for the contact problem.

Each of the LSTM boxes in the diagram represents one of the LSTM cell of Figure 1. Each LSTM cell has n number of hidden units. The number of hidden units in the LSTM cell represents the degree of decomposition to which the incoming time sequence will be mapped. For example, an LSTM with 30 hidden units will decompose the two-dimensional input vector into 30 time sequences. Figure 5 shows the transformation of the input through every layer. The last layer shows the prediction (orange) and the actual response (black). In this example, the LSTM takes in two time sequences and transforms them into 30 time sequences. The LSTM then takes the input transformed into a high-dimensional space through a series of nonlinear operations, until they are combined into a single time sequence in the last layer. The transformations are determined by the weights of the hidden units. Each gate has different weights and they control the flow of information, as determined by Equations (2-6). These weights are applied to the input over the entire length (time wise) of the input. Since the input is 2 dimensional and the number of hidden weights is 30, the LSTM effectively maps the input from 2 to 30 dimensions. The input weights were represented by W in the equations above. The rest of the weights were represented by U. Physically, one could think of the high-dimensional representation of the input as representing a change of basis. In this interpretation, the program is able to generalize when it finds the proper basis representation of the signal, and the proper algorithm that transforms such basis representation into the output. There is no further compression or decomposition going from layer 1 to layer 2, hence the weight matrix is square. A desirable

property of the weight matrices is sparsity. Sparse weight matrices may indicate a reduction in redundancy in the feature representation, and consequently, better utilization of the network capacity [24]. Though no explicit effort has been taken to enforce sparsity, the matrices in Figure 5 show sparsity. This sparsity is likely a consequence of dropout.

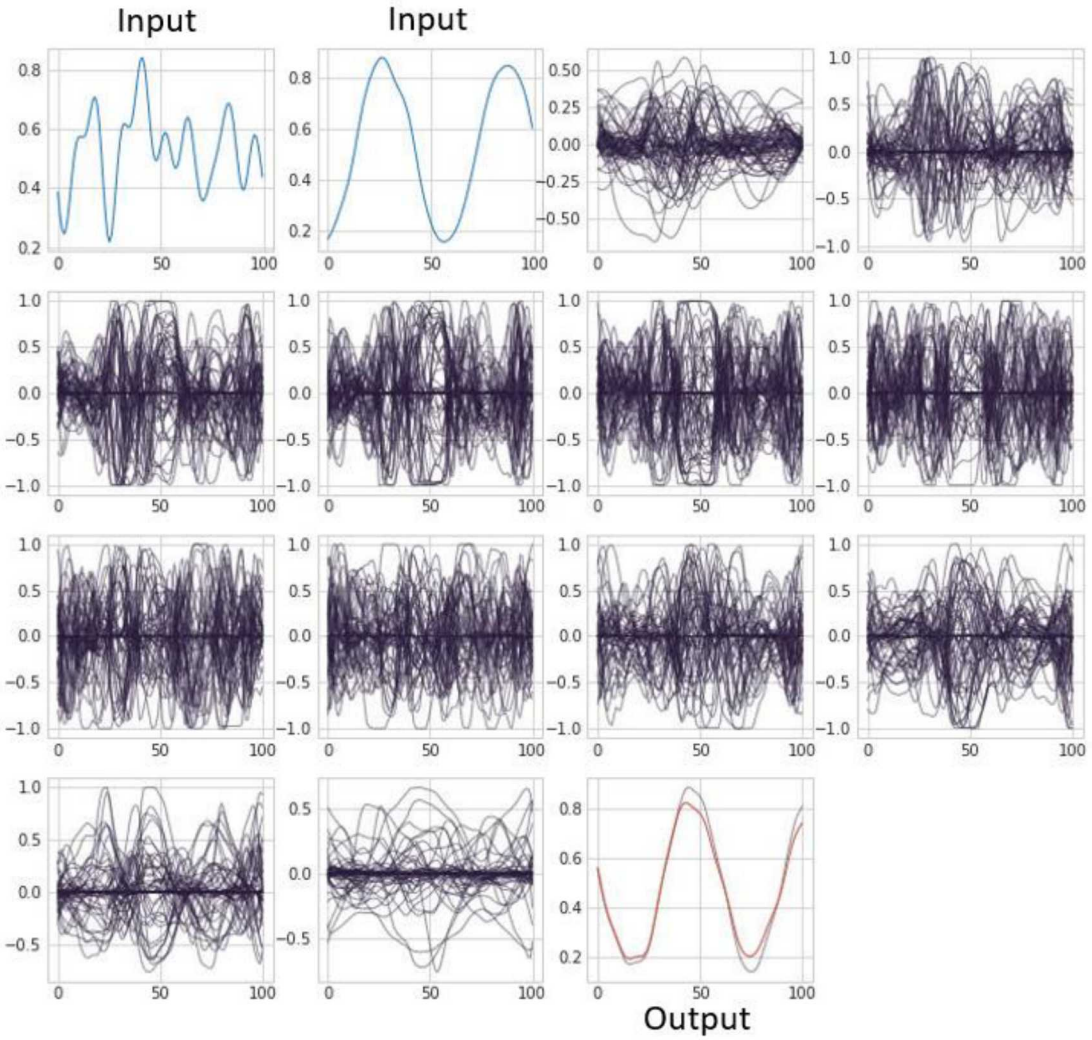

Figure 5. The transformed input signals as they traverse the depth of the LSTM network.

3.1.2 Results

The full network was trained using synthetic data generated in Matlab by solving the three DOF contact problem. Several time series realizations of a random vibration excitation were generated from a single PSD using the method of random phase. The time signals share statistical properties, but they are unique realizations. The network was trained on one of the realizations and tested on a completely different realization, called the test dataset.

Figure 6 shows the input random excitation (top) and the corresponding response (bottom). The response from direct numerical time integration is shown in red, and the neural network prediction is plotted in black. The initial data used to seed the prediction is shown in blue in the bottom plot. The blue portion of the bottom plot is the only information needed to perform the prediction after the network has been trained. It

can be seen that the overall shape of the response is captured by the prediction. Figure 7 shows a close-up of the prediction. As illustrated, the prediction is performed stochastically by performing a Monte Carlo simulation. In this case, 20 realizations of the response are shown in black. The variance in the predictions is a way to quantify the model-form error, as discussed in Section 2.2.

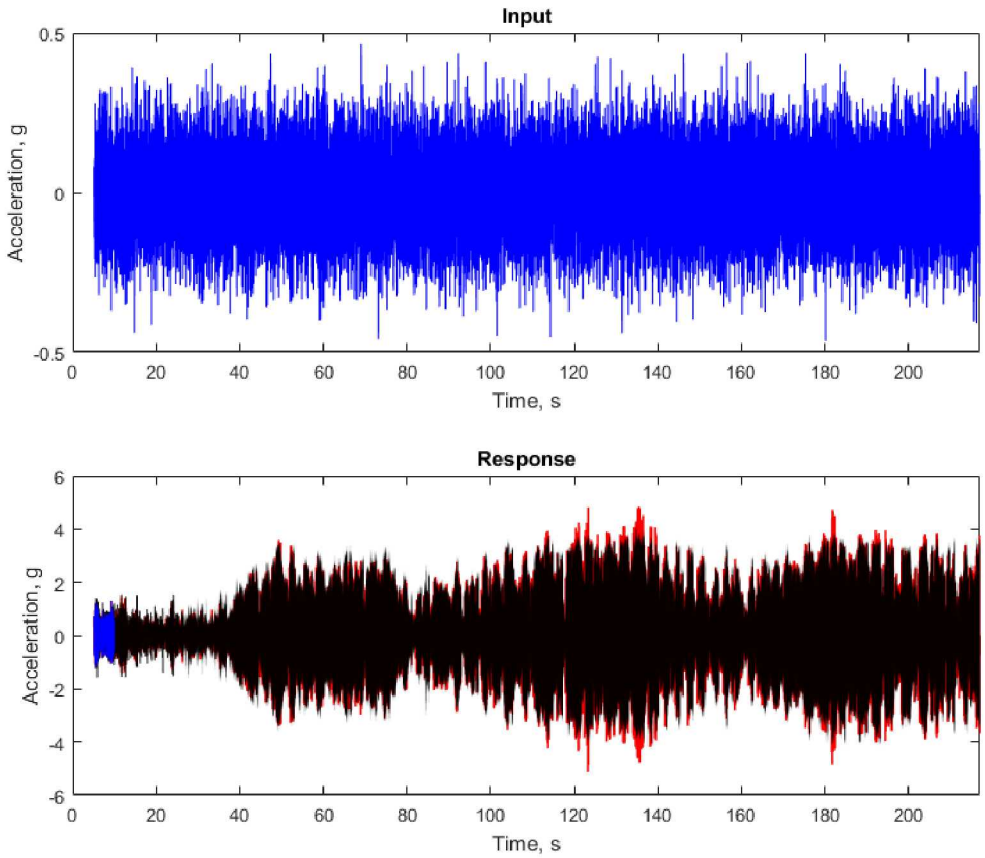

Figure 6. Input random vibration excitation (top) and response (bottom). In the bottom image, the actual response is shown in red, the network prediction is shown in black, and the "seed" use to generate the prediction is shown in blue.

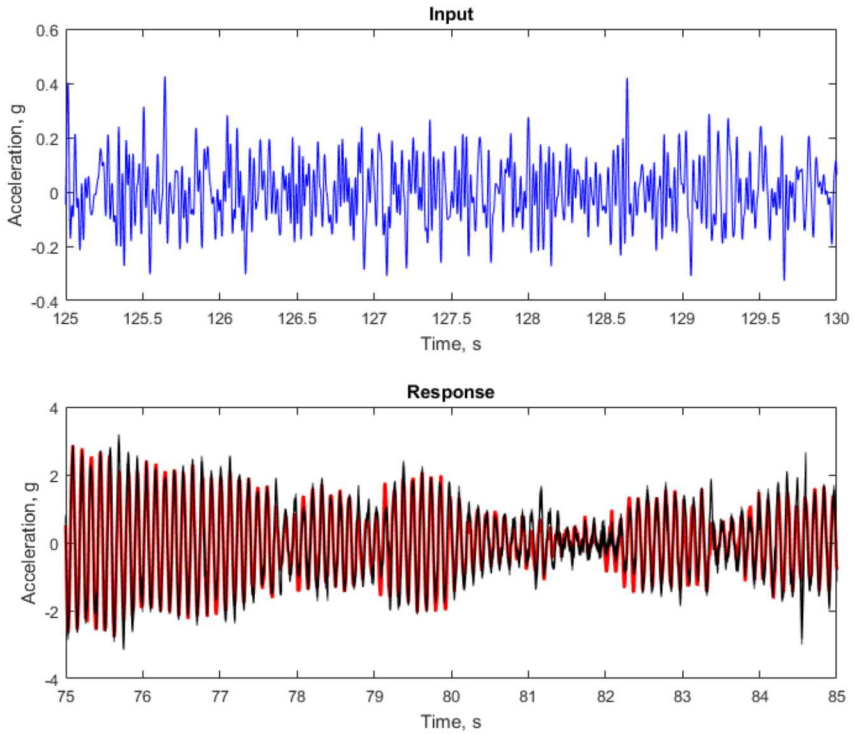

Figure 7. Close-up of input random vibration excitation (top) and response (bottom). In the bottom image, the actual response is shown in red, the network prediction is shown in black, and the "seed" use to generate the prediction is shown in blue.

Figure 8 shows the same comparison of PSDs in the frequency domain. The prediction envelopes the actual response and it matches the peaks well. Figure 9 shows the comparison using a spectrogram calculated with the Short-Time Fourier Transform. The spectrogram also shows a good match between the actual response and the prediction. The error is larger at lower frequencies. This error may be a consequence of the window size chosen to partition the signal in Section 3.1.1. A longer window would capture lower frequency content better but would decrease the number of training samples. More work is needed to determine an optimal window size.

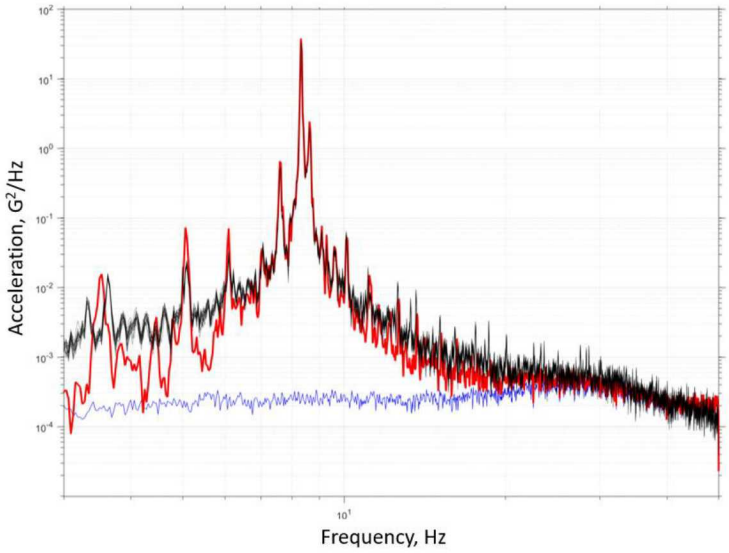

Figure 8. Power spectral density of input (blue), actual response (red), and prediction (black).

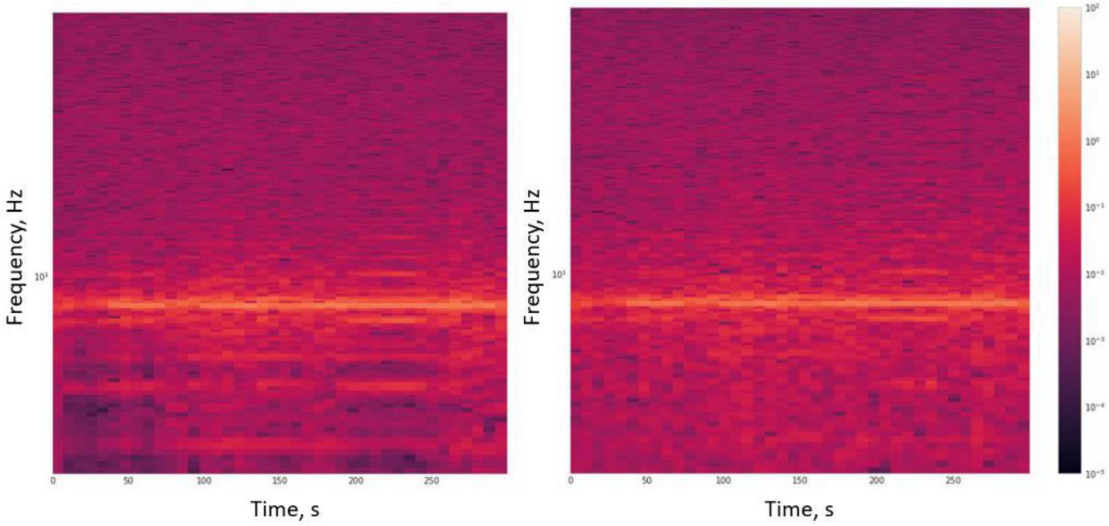

Figure 9. Spectrogram of actual response (left) and network prediction (right).

Lastly, the model parameters (K and M) were varied to study the effect of parametric uncertainty. The parameters were varied +/- 10% of their nominal value to model the uncertainty associated with these values. A few values were sampled from a uniform distribution in the +/- 10% range from the nominal values and their responses were evaluated with the Maltab model. A short "seed" from each evaluation was used to predict the long-term response using the LSTM network. Figure 10 shows the comparison between the actual response and the prediction. For the most part, the network is robust enough to capture small parametric variations. The effect of the change of parameters is embedded in the "seed" data, which the network is able to identify and accurately capture the evolution of the response. The multiple predictions correspond to different model realizations due to the network having been trained with dropout. The multiple realizations are an expression of model-form uncertainty.

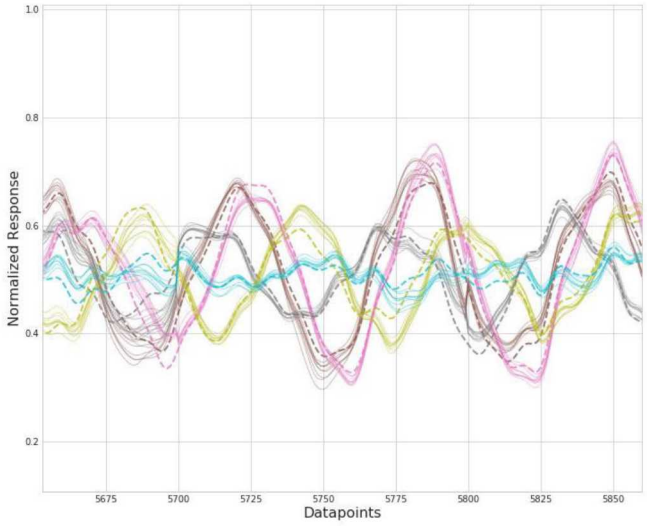

Figure 10. The response of five variations of the model parameters shown. The actual responses are shown with a dashed line and the predictions are shown with solid lines.

3.2 The Ministack

The Ministack assembly is a benchmark specimen that was developed at Sandia National Laboratories (SNL) to better understand the nonlinear damping and stiffness of a compression fit interface. The structure consists of an outer can that houses a subassembly of an aluminum slug that fits tightly in between two foam cups. For this work, a reduced-order model of the Ministack that was developed by Kuether and Najera [25] was used to generate training data. The reduced order model includes a nonlinear interface represented by an Iwan element. The Iwan element consists of a constitutive model that captures the microslip behavior and nonlinear dependence of damping and stiffness, as the amplitude of the response increases [26]. The Iwan element parameters corresponding to the Ministack frictional interface were found through optimization in [25]. *Figure 11* shows the Ministack finite element model and the frictional interfaces that were reduced to Iwan elements. The random vibration input was applied at the base of the can and the response was recorded at the top of the slug.

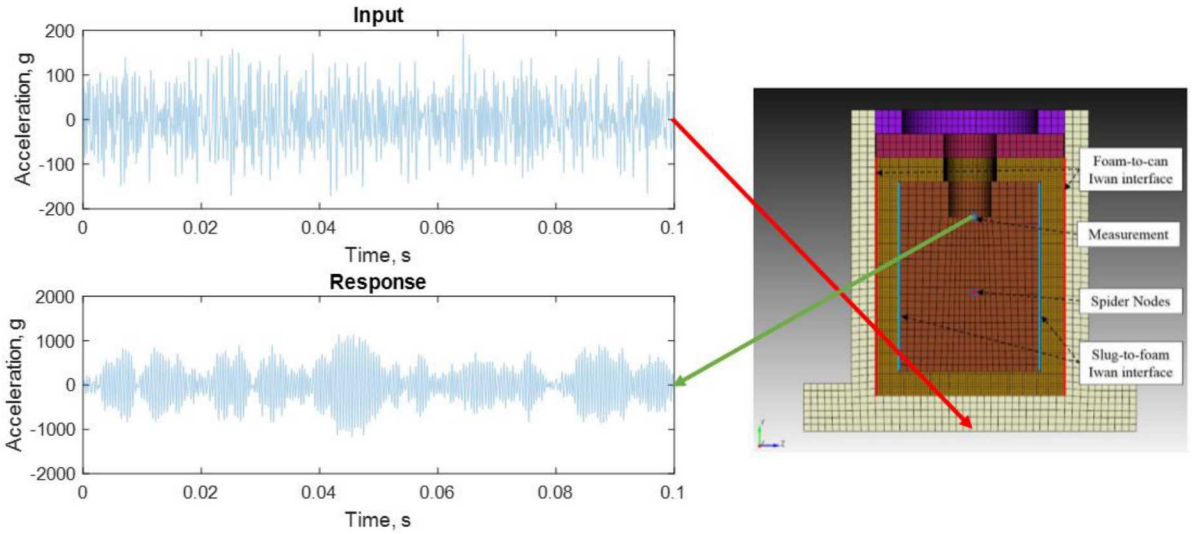

Figure 11. The Ministack finite element model with the interfaces that were reduced to Iwan elements with the random vibration input and response locations indicated.

3.2.1 LSTM Architecture

The LSTM architecture used to model the Ministack problem was a deep LSTM network with dropout layers, and a fully connected output layer. The architecture used was similar to that of the contact problem described in Section 3.1.1. One of the differences was that the Ministack LSTM was not bidirectional. It was found that the Ministack could be modeled without including bidirectionality. Another difference was the output layer. In the case of the contact problem, the output layer was time distributed while that was not the case for the Ministack. It was found that a static dense layer with multiple features representing the time dimension worked best for the Ministack. The exploration of the inner workings of the Ministack LSMT are not provided here but similar conclusions to the contact problem can be drawn. The LSTM network finds the appropriate basis representation of the signal, and the graph structure that best performs the mapping of input to output.

3.2.2 Results

The network was trained using synthetic data generated in Matlab using the reduced-order model described in [25]. As in the contact problem, the results provided in this section were generated using the test dataset.

Figure 12 shows the input random vibration (top) and the response (bottom). In the bottom plot, the actual response is shown in red, the prediction is shown in black, and the seed data used to generate the response with the LSTM network is shown in blue. The predicted and actual responses are nearly indistinguishable. Figure 13 shows a close-up of the data to illustrate the close match between prediction and actual. Since the LSTM network provides stochastic predictions, the prediction shown in Figure 13 represents the mean prediction.

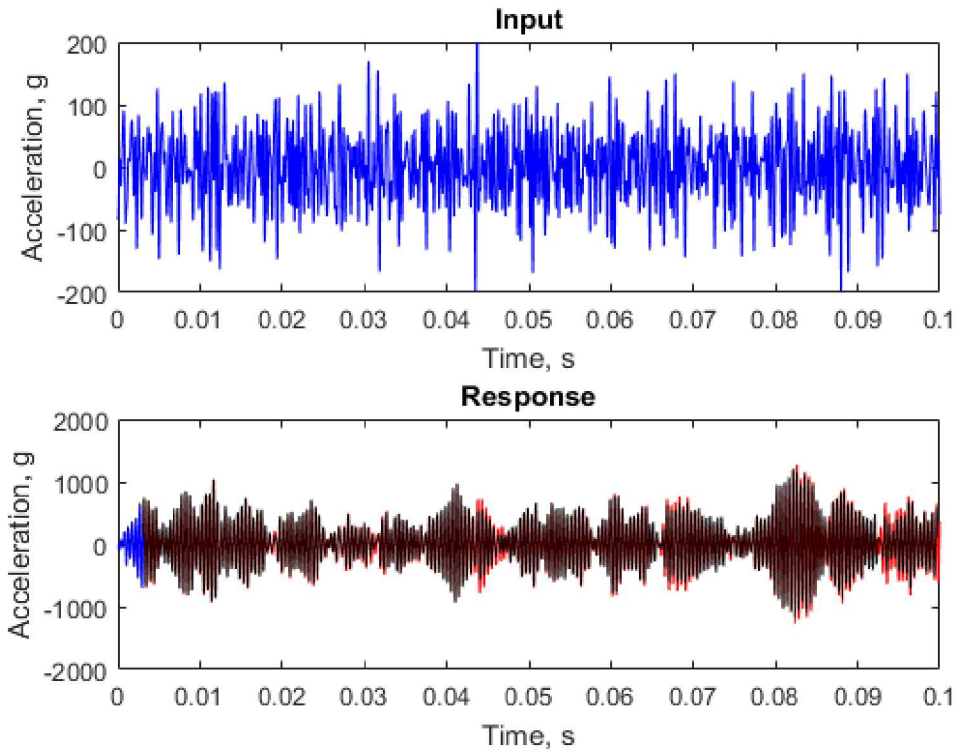

Figure 12. Input random vibration excitation (top) and response (bottom). In the bottom image, the actual response is shown in red, the network prediction is shown in black, and the "seed" use to generate the prediction is shown in blue.

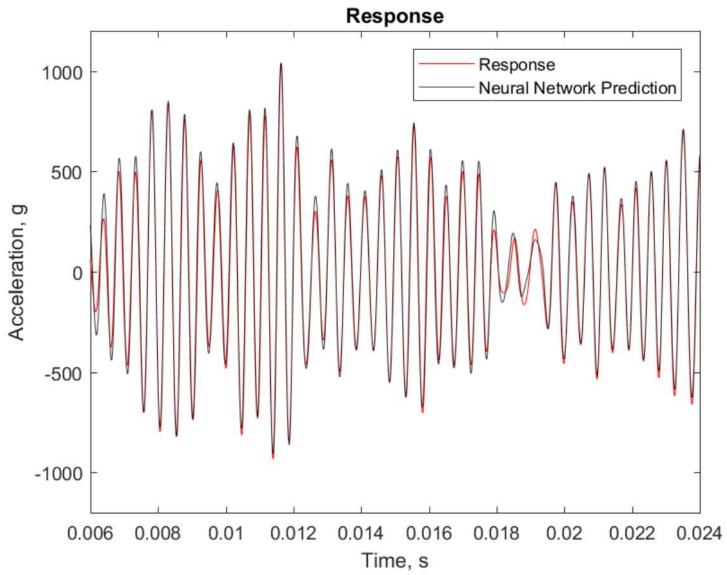

Figure 13. Close-up of Ministack actual response (red) and LSTM network predictions (black).

To illustrate the stochastic predictions of the LSTM network with dropout, Figure 14 shows multiple realizations of the prediction in black. Variance in the response can be used to quantify the epistemic uncertainty of the response, as described in Section 2.2. Uncertainty, represented by the range of predictions, tends to be larger in areas where the LSTM network error is larger (e.g., in areas the network did not encounter during training).

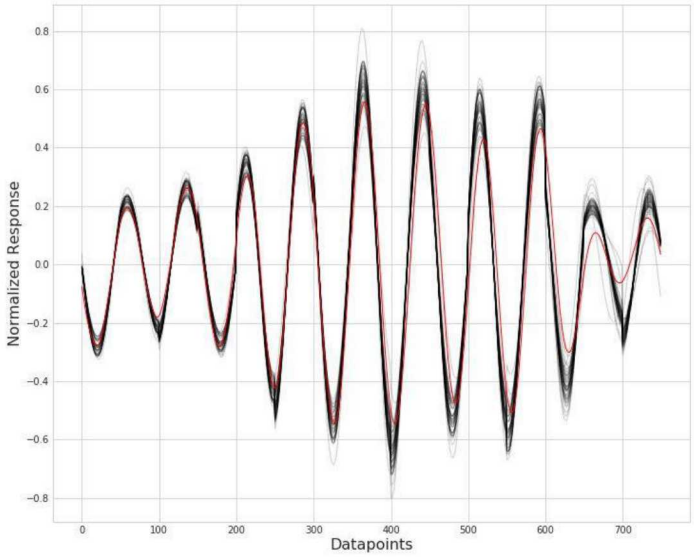

Figure 14 - Response including error

4 Conclusion

This paper presents a novel method using Long Short-Term Memory neural networks to predict the response of nonlinear systems subject to random broad-band excitation. By using the LSTMs ability to use past as well as future data in its prediction, accurate predictions of strong and weak nonlinearities are possible. Two typical nonlinearities in the form of intermittent contact and frictional sliding were used as baseline

examples, both of which were shown to be solvable with the LSTM neural network. Additionally, it was shown that by including dropout layers, the model becomes stochastic in form allowing it to predict epistemic and aleatoric uncertainty inherent to the analysis.

Acknowledgements

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] P. H. Wirsching, T. L. Paez and K. Ortiz, Random Vibration Theory and Practice, New York: John Wiley and Sons, Inc., 1995.

[2] T. L. Paez, "Random Vibration - A Brief History," Sound and Vibration, January 2012.

[3] J. Shoneman, C. Ostoich, L. Jarman, C. Van Damme and M. S. Allen, "Linear and Nonlinear Reduced-Order Modeling for Hypersonic Panel Flutter," AIAA SciTech Forum, 2018.

[4] C. R. Hua, Y. Zhao, Z. W. Lu and H. Ouyang, "Random Vibration of Vehicle with Hysteretic Nonlinerar Suspension under Road Roughness," Advances in Mechanical Engineering, vol. 10, no. 1, pp. 1-10, 2018.

[5] M. R. Brake, The Mechanics of Jointed Structures, Springer International Publishing, 2018.

[6] S. H. Crandall, "Perturbation Techniques for Rnadom Vibration of Nonlinear Systems," The Journal of the Acoustical Society of America, vol. 35, no. 11, 1963.

[7] J. B. Roberts and P. D. Spanos, Random Vibration and Statistical Linearization, New York: Wiley, 1990.

[8] G. K. Er, "Multi-Gaussian Closure Method for Randomly Excited Nonlinear Systems," International Journal of Nonlinear Mechanics, vol. 33, no. 2, 1998.

[9] R. Z. Khasminskii, "A Limit Theorem for the Solution of Differential Equations with Random Right-Hand Sides," Theory of Probability and its Applications, vol. 11, no. 3, 1966.

[10] H. Risken and F. Till, The Fokker-Planck Equation: Methods of Solution and Applications, Berlin: Springer-Verlag, 1996.

[11] S. Narayanan and P. Kumar, "Numerical Solutions of Fokker-Planck Equations of Nonlinear Systems Subjected to Random and Harmonic Excitations.," Probabilistic Engineering Mechanics, vol. 27, pp. 35-46, 2012.

[12] R. Vargas, A. Mosavi and L. Ruiz, "Deep Learning: A Review," Advances in Intelligent Systems and Computing, 2017.

[13] S. Haykin, Neural Networks, A Comprehensive Foundation, 2nd ed., Upper Saddle River: Prentice-Hall, Inc., 1999.

[14] S. Hochreiter and J. Schmidhuber, "Long Short Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[15] S. Hockreiter, Y. Bengio, P. Frasconi and J. Schmidhuber, "Gradient Flow in Recurrent Nets: The difficulty of Learning Long-Term Dependencies," A Field Guide to Dynamical Recurrent Neural Networks, 2001.

[16] A. Graves, A.-R. Mohamed and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, 2013.

[17] A. Graces, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke and J. Schmidhuber, "A Novel Connectionist System for Improved Unconstrained Handwriting Recognitions," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 855-868, 2009.

[18] T. Luong, I. Sutskever, Q. V. Le, O. Vinyals and W. Zaremba, "Addressing the Rare Word Problem in Neural Machine Translation," arXiv: 1410.8206 [cs], 2014. [Online]. Available: https://arxiv.org/abs/1410.8206.

[19] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenkno and T. Darrell, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description arXiv: 1411.4389," 2014. [Online]. Available: https://arxiv.org/abs/1411.4389.

[20] C. Olah, "Colah's Blog," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[21] W. Zaremba, I. Sutskever and O. Vinyals, "Recurrent Neural Network Regularization arXiv:1409:2329," 2015. [Online]. Available: https://arxiv.org/abs/1409.2329.

[22] N. Srivastave, G. Hinton, A. Krizhevsky, S. Ilya and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Leanring Research, vol. 15, pp. 1929-1958, 2014.

[23] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in Proceedings of the 33rd International Conference on Machine Learning, New York, 2016.

[24] S. Changpinyo, M. Sandler and A. Zhmoginov, "The Power of Sparsity in Convolution Neural Networks arXiv:1702.06257," [Online]. Available: https://arxiv.org/pdf/1702.06257.pdf.

[25] R. J. Kuether and D. A. Najera, "Modeling Nonlinear Energy Dissipation of the Ministack Assembly (SAND2017-10517)," Sandia National Laboratories, Albuquerque, 2017.

[26] D. J. Segalman, "A Four-Parameter Iwan Model for Lap Type Joints," Journal of Applied Mechanics, vol. 72, no. 5, pp. 752-760, 2005.