**Proceedings of the ASME2021
48th Annual Review of Progress in
Quantitative Nondestructive Evaluation
QNDE2021
July 28-30, 2021, Virtual, Online**

QNDE2021-75080

ESTIMATING GUIDED WAVE VELOCITY VARIATION WITH NEURAL NETWORKS

Ori Leibovici[1], Kang Yang[1], and Joel B. Harley[1]

[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL

ABSTRACT

While guided wave structural health monitoring (SHM) is widely researched for ensuring safety, estimating performance deterioration, and detecting damage in structures, it experiences setbacks in accuracy due to varying environmental, sensor, and material factors. To combat these challenges, environmentally variable guided wave data is often stretched with temperature compensation methods, such as the scale transform and optimal signal stretch, to match a baseline signal and enable accurate damage detection. Yet, these methods fail for large environmental changes. This paper addresses this challenge by demonstrating a machine learning method to predict stretch factors. This is accomplished with feed-forward neural networks that approximate the complex velocity change function. We demonstrate that our machine learning approach outperforms the prior art on simulated Lamb wave data and is robust with extreme velocity variations. While our machine learning models do not conduct temperature compensation, their accurate stretch factor predictions serve as a proof of concept that a better model is plausible.

1 INTRODUCTION

Structures, such as planes or buildings, are susceptible to defects and damage over time. The goal of structural health monitoring is to monitor these structures and the damage evolution over time [1]. To maintain the safety of individuals who use and interact with these structures, research has been conducted to create structural health monitoring (SHM) systems and techniques to perform *in situ* nondestructive evaluation (NDE) [2].

Guided wave ultrasonics (GWU) has become an attractive monitoring technique due to its ability for damage diagnosis at large scales in plate structures [3, 4]. GWU is common in NDE due to its long range diagnostic capabilities with limited attenuation [5–7]. As illustrated in Figure 1, a transmitting sensor creates ultrasound waves that propagate through a material and reflects off surfaces and damage within the material. A receiving sensor then measures the signal and saves the data to memory in a computer. Advanced SHM algorithms can be applied to the received data and perform numerous NDE oriented tasks, such as damage detection [8].

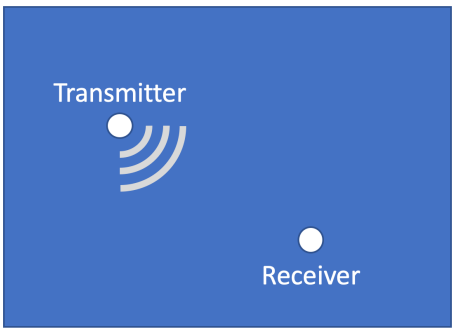


FIGURE 1: ILLUSTRATION OF ULTRASONIC GUIDED WAVES PROPAGATING THROUGH A MATERIAL FROM A TRANSMITTING SENSOR AND READ AT A RECEIVING SENSOR

Guided wave SHM has setbacks due to environmental, sensor, and material factors. One of the most notable challenges is changes in temperature [8, 9]. Temperature predominantly changes the velocity of the ultrasonic signal [5] and makes the prediction of damage and other features challenging [3]. Guided waves are often more sensitive to changes in temperature than material damage [5, 8]. To combat this sensitivity, environmentally variable sensor data can be transformed to match a baseline signal and enable accurate damage detection [9, 10]. Such methods for performing temperature compensation on guided waves include the scale transform [5] and optimal signal stretch [9–11]

methods. These techniques stretch the guided wave data to emulate a change in velocity. By identifying the stretch factor that minimizes the error between the two signals, the methods can identify the velocity change between them. It has been shown that with proper baseline subtraction with temperature compensation, damage detection accuracy is greatly improved [5, 9–11].

The state-of-the-art in stretch based temperature compensation is the scale transform due to its quick computation speed when compared to other methods [5]. The scale transform operates on signals in the stretch factor domain. The scale transform assumes a time-stretch has occurred between two signals due to changes in velocity, resulting from variations in temperatures [5]. Yet, as the stretch factors reach extreme values, performance declines. This is because a stretch only approximates a velocity change for low stretch factors. This leads to inaccurate velocity estimates and reduces damage detection performance [3].

Note that the definition of an "extreme" stretch factor depends on many conditions, including the signal bandwidth, center frequency, travel distance, and number of reflections (i.e., the diffuseness) of the waves [12]. As a result, bounds of performance can be difficult to estimate. In addition, temperature also effects the attenuation and dispersion of Lamb waves [5]. These effects are not accounted for in temperature compensation models and provide more mismatch.

Such limitations call for a new approach. In this paper, we utilize machine learning due to its high success rate in modeling complex functional relationship, including in NDE [13]. Hence, machine learning enables us to process Lamb waves despite their complex, mutli-modal, and dispersive wave behavior. Machine learning algorithms have the capability to learn features from training data, make accurate predictions, and see patterns in new data [14, 15]. Sufficiently large data sets are required but is attainable with mass consumption of sensor data over long periods of time and through simulations. Modern computers have the capability to hold large data sets in memory and to run machine learning algorithms.

Machine learning has been successfully used in SHM on ultrasonic guided waves to detect and predict damage [16]. With guided waves, specifically, machine learning models have demonstrated a capability to achieve fast and automated damage detection with accuracies of 90 percent or higher, achieving performance higher than other state-of-the-art techniques [15]. To date, most machine learning algorithms have focused on damage detection and damage localization problems. Such research implies that machine learning can have a reach into other SHM domains such as temperature compensation.

Due to the high variability in ultrasonic waves, machine learning techniques can be successfully used to predict desired material features. The neural network, a machine learning model, can tackle such a task through its ability to act as a universal approximator [17]. Prior unknown information is hidden in data and the neural network can capture this information through training [17]. Training is done over a large data set and known labels, the desired output values. After many iterations, the neural network model is able to predict the desired labels. There are many different models each having applications for different problems [17].

To estimate velocity variations at extreme values, we created two machine learning models. The labels are defined as changes to the ultrasound wave velocity resulted from changes in temperature. The models created for this work are both feed-forward neural networks (FFNN). We use FFNNs due to their ability to accurately approximate complex, continuous functions [17]. Note that this work only predicts stretch factors / velocity estimates. It does not perform temperature compensation, which is a topic for future work. If a machine learning model can more accurately predict stretch factors than the prior art, it can be concluded that a better machine learning inspired temperature model is plausible.

In Section 2 a brief background of neural networks is given to aid in the scope and context of this work. In Section 3 we describe our simulation data, neural network frameworks and architecture, and training algorithms. We then present our stretch factor prediction results compared to the scale transform prior art in Section 4.

2 BACKGROUND OF NEURAL NETWORKS

Machine learning is a class of methods for automated data analysis and model creation. Machine learning is based on the artificial intelligence philosophy that computer systems can learn and identify patterns in data without human intervention [18]. Machine learning models have been better interpreters of data than humans. Neural Networks (NN) are a subset of machine learning and with numerous layers for learning. Neural networks with many layers form the basis for deep learning.

Neural networks are loosely inspired by neurons from the human brain in which each neuron holds data and sends information to numerous other neurons in a directed graph architec-

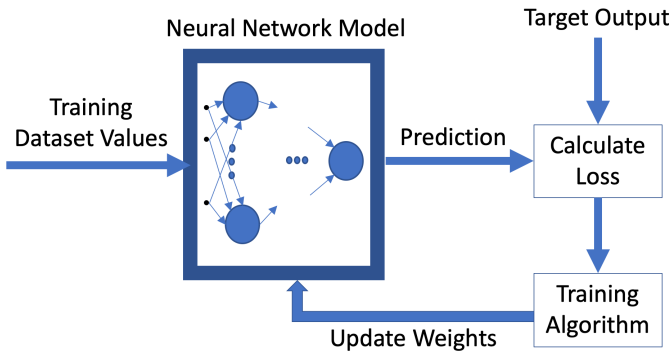


FIGURE 2: NEURAL NETWORK DURING LEARNING

ture [17]. FFNNs are a type of neural network where connections between nodes do not form a cycle. FFNNs can have numerous layers consisting of an input layer, one or more hidden layers, and an output layer. In the case of FFNNs, each neuron from a layer attaches to each neuron of the next layer by connections called weights [17].

$$f(b+\sum_{i=1}^{n} x_i y_i) \tag{1}$$

Equation 1 shows each neuron's value, starting with the second layer onward, which is the sum of a bias b and the summation of the products of all the previous neurons values x_i and their respective connecting weights w_i. The resulting value is then passed through an activation function, $f(\cdot)$, which conforms the result to be within a range of desired values. Each neuron has its own set of weights and bias that connect to it from the previous layer. Neuron values can be passed through other functions like batch normalization, if needed. Neurons follow this computation and connection pattern until a final set of one or more value(s) is collected at the output layer. The first layer of a FFNN takes in input data. FFNNs can have numerous layers and use different activation functions to fit a problem's needs [17].

In order to characterize a problem's high dimensional behavior to output accurate predictions, the process of training must occur on an FFNN. Training is the process of repeatedly inputting data, from a designated dataset, into the FFNN and constantly modifying biases and weights of all neurons through a method called backpropagation [19]. Backpropagation works by computing the gradient of a chosen loss function by comparing correct output values, called labels, to the predictions of the FFNN and corrects the values of the respective weights and biases through gradient descent [20]. Stochastic gradient descent is done on subsets of the input data, called batches, to increase efficiency of learning, whereas, gradient descent uses the entire dataset [21]. Stochastic gradient descent makes approximations for the new values of weights and biases, but over many iterations does not sacrifice efficiency [20, 21].

The process of training is illustrated in Figure 2, where designated training dataset values are passed into the FFNN. All neuron values are computed for each layer, passing through the computations from (1). The output(s) are collected and compared to labels, the loss is calculated with a loss function, and weights and biases of the model are updated using training, which includes backpropagation [22]. This process is repeated until the predictions reach a desired accuracy [22]. The processes just described have a powerful effect of modeling high dimensional system behavior to high accuracy, even on data not seen before as patterns within data shape the NN function. Gradient descent brings the NN weight and bias values to a subspace

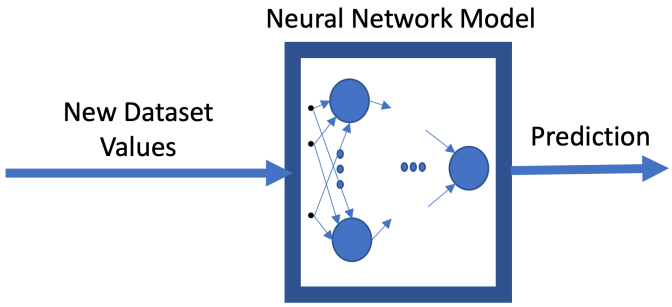


FIGURE 3: NEURAL NETWORK DURING TRAINING, TESTING, AND USE

defining a high dimensional local minima [23].

Once training is completed, validation is applied by showing the FFNN new data it has not seen before. The FFNN model parameters are repeatedly tweaked until the validation data predictions are desirable. Lastly a dataset, separate from training and validation, is used to test to see if the model's outputs still perform well through a process called testing. If desirable outputs are achieved from testing, it can be accepted that the FFNN performs well [24]. The FFNN can then be used in the field to make accurate predictions for the task assigned. The machine learning tasks described here are a form of supervised learning [25].

The flow of training, testing, and use is illustrated in Figure 3. Separate data from learning is passed into the model and a prediction is collected. If training is conducted, the process of updating model parameters and training will be done until predictions are desirable. Otherwise, FFNN outputs can be collected for testing or use in the field.

3 METHODS

This section discusses the data we generate to estimate velocity changes as well as the neural networks we train with this data.

3.1 DATA

Our dataset has 50,000 simulated ultrasonic guided Lamb wave pairs, created from the Rayleigh-Lamb equation. Each pair of waves contains a baseline and a "stretched" baseline with a different velocity, as seen in Figure 4. Lamb waves propagate in solid plates. They are elastic waves whose motion lies with the particles in the material. Each guided wave signal is 2,500 data points, representing 2.5ms of data. The velocity stretch factor, s,

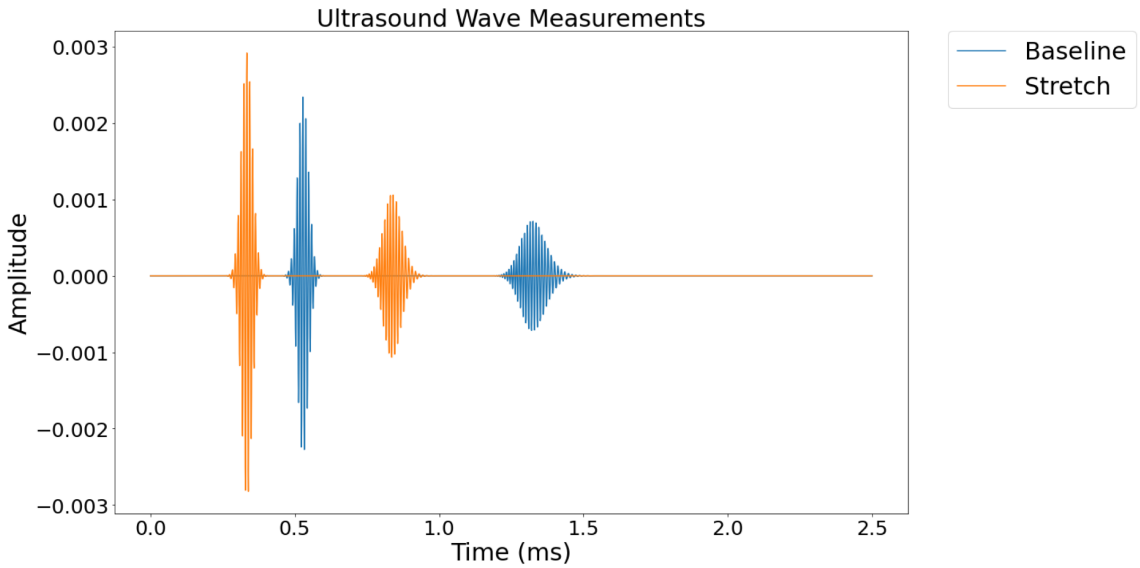


FIGURE 4: INPUT DATA TO FFNN

is ranged from 0.5 to 1.5 times the baseline wavenumber

$$k_{\text{new}} = s\,k \tag{2}$$

which can be understood as a change in velocity based on

$$k = \frac{\omega}{v} \tag{3}$$

where k is wavenumber, v is velocity, and ω is angular frequency. These velocity stretch factors were saved as labels (the output of the FFNN). The first 40,000 measurements were used for training, and the last 10,000 were used for testing.

The baseline signals wavenumber is k and the "stretched" signals wavenumber is $s\,k$. This change in velocity represents how changing temperatures change a signal in a material [5]. The difference between the two signals has more complexity than a time stretch done by the scale transform, as seen in Figure 4. The change in velocity does indeed cause the signal to arrive later, but the late arrival also accounts for attenuation. The changing velocity lastly affects Lamb wave dispersion, seen in Figure 4, where the time-varying frequency behavior changes after the velocity change [5].

The two excitations in each signal account for the S0 (zeroth symmetric) mode and A0 (zeroth asymmetic) mode. As ultrasound waves are excited in a plate, different modes result, each having the same frequency and a fixed phase relation. The modes are characterized as sinusoidal motion [26]. The simulations were designed to have limited reflections, which makes the stretch based temperature compensation methods less effective, as they have superior performance in diffuse wavefields.

3.2 FFNN MODELS

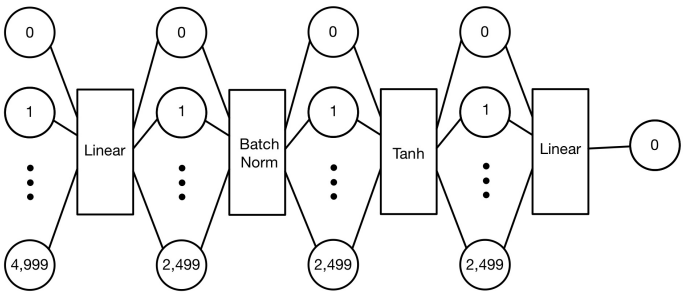
The machine learning framework Pytorch was used to train the machine learning task of predicting velocity stretch factors in ultrasonic wave measurements of a plate with a training algorithm and two FFNN models. The first model was a 3-layer model with tanh activation functions, as seen in Figure 5a. The second model omits the activation functions and instead applies a natural log to the input data and an exponential operation after hidden layers, as seen in Figure 5b. Both approaches were constructed the same, besides the activation function, for direct comparison. The model structure and learning algorithms were influenced using tutorials from [27].

The tanh model was used as it is in the standard construction of a FFNN [28]. A second model was chosen so that data could mix in the logarithm space, where multiplication becomes addition.

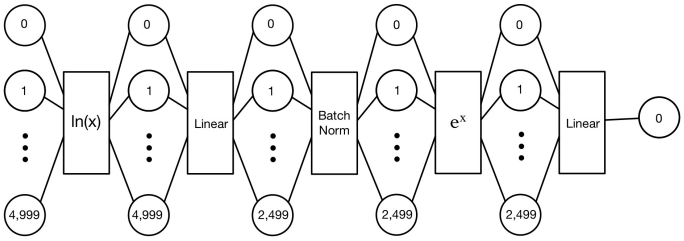
$$\log(a\,b) = \log(a) + \log(b)\ . \tag{4}$$

The mixture of logarithm and exponential operation enables the neural network to learn how to compute dot products. We chose this framework since the dot product is used for many algorithms for signal comparison. It is more difficult for nonlinearities, like activation functions, to compute the solution to the inner products alone [29]. The data was normalized between the largest and smallest value before entering as logarithm cannot take zero or negative numbers.

The models' input was of size 5000, the hidden layer was of size 2500, and the output was of size 1. The input size equates to each pair concatenated as each was of size 2500. The single out-



(a)



(b)

FIGURE 5: (a) TANH FFNN (b) LOGARITHM FFNN

put is the stretch factor prediction. Each model used 500 epochs, meaning they ran through the whole data for each epoch. They had a batch size of 100, as stochastic gradient descent is used. They used a mean squared error loss function, and a learning rate of 10^{-5}, which is how large steps are taken during stochastic gradient descent, seen in Table 1.

Batch normalization was used as seen in Figures 5a and 5b to standardize each batch, creating consistent mean and variance, allowing for correct predictions [30]. Batch normalization is a technique that aims at improving training by stabilizing the distribution of layer inputs [30]. Introducing 1D batch normalization allowed for re-centering and re-scaling adjustment of the input layer, exactly what each model needed to function. It is believed that batch normalization works by decreasing internal covariate shift, which has a negative impact on training [30]. Overall, the models would not train properly without batch normalization because each training batch was very different from each other. Without it, the models predictions would converge to the average and "central prediction" as each batch moves the model weights in different directions, making the correct predictions a moving target.

4 RESULTS AND DISCUSSION

The testing data, separate from the training data (note: our setup had no validation step), was applied to both FFNN models as well as with the scale transform to estimate the stretch factors

Training Specifications	
Model	Feed Forward
Epochs	500
Batch Size	100
Inputs	50,000
Input Size	5,000
Hidden Dimension	2,500
Output Size	1
Criterion	Mean Squared Error Loss
Learning Rate	$1.00 * 10^{-5}$

TABLE 1: NEURAL NETWORK TRAINING SPECIFICATIONS

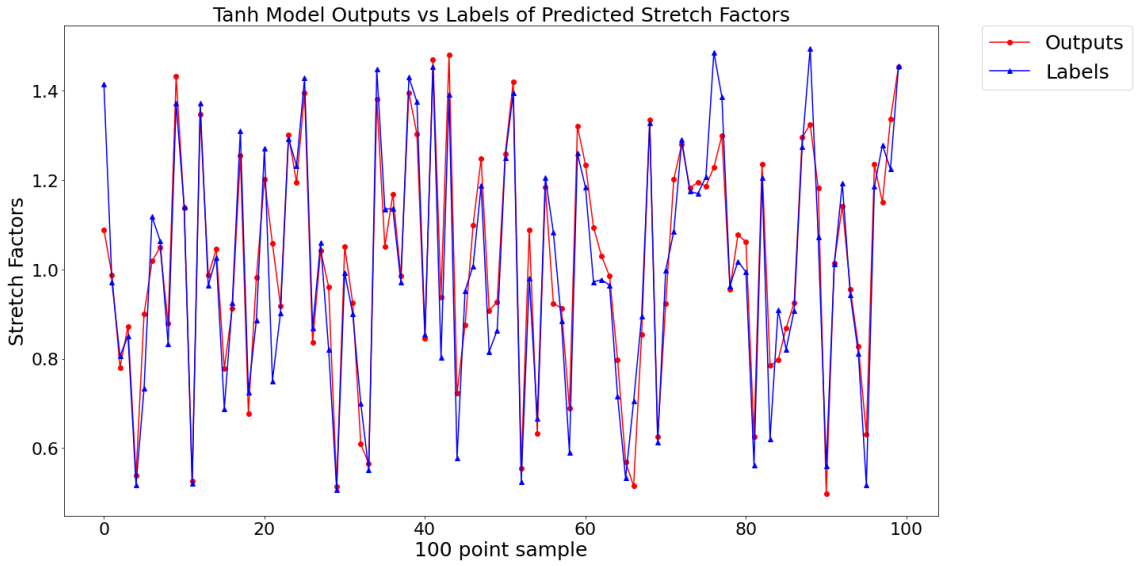
and compare results. Figure 6a and figure 6b plot the FFNN output predictions compared to the stretch factor labels. The figures display a 100 point sample, the values are not related to each other and are random stretch factor values. The values are connected for improving the visual comparisons. It can be seen that both models did well, but the log model performed better in Figure 6b. An example, among others, is the left most prediction, where the log model performs significantly better.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2 \tag{5}$$

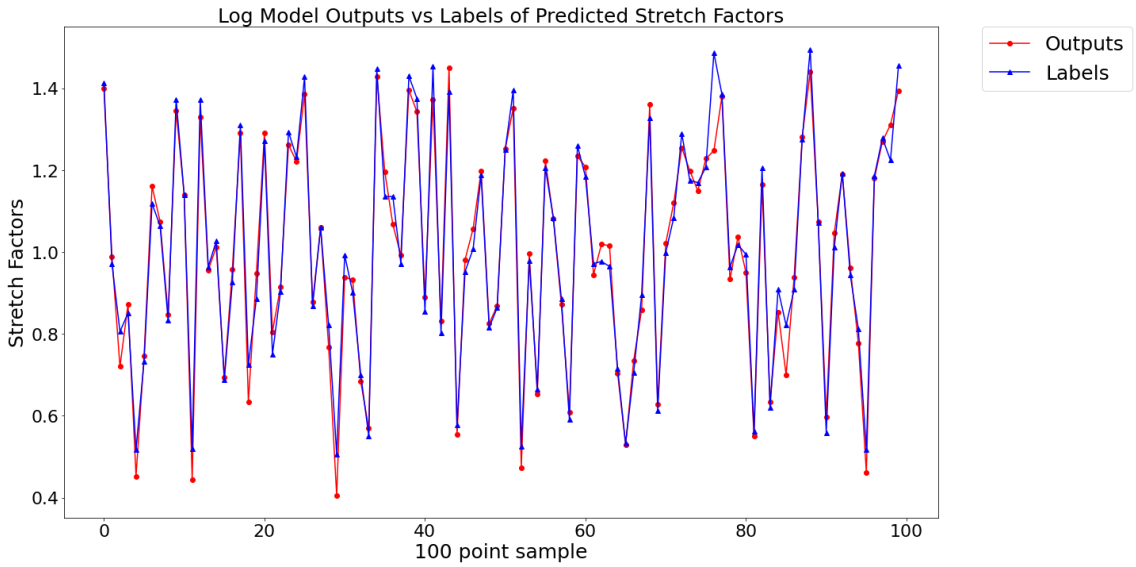
Mean squared error (MSE) was used as a measure of performance across the models. MSE measures the average squared error and corresponds to the expected value of the squared error loss [31]. MSE is used to evaluate how close the models' predictions were to the actual values. As it is the second moment of the error, it incorporates the variance and bias of the estimator. With this, MSE gives the spread of the values and how far they are from the true values [31]. In (5), n refers to the size of the testing data. X and Y refer to the FFNN prediction and label values.

From our predictions, the scale transform MSE was 0.0525, the standard FFNN MSE was 0.0084, and the logarithm FFNN MSE was 0.0037, shown in Table 2. The log FFNN model MSE overcame the scale transform MSE by a factor of **14.2** and the tanh FFNN model MSE by a factor of **2.25**. It is believed that the natural logarithm performs better because the logarithm transformation changes multiplication to addition and enables dot products to be computed with a neural network, from Equation 4 [29]. With further experimentation with hyperparameters, both models' performance can be optimized, yielding even higher prediction accuracy [32].

Figure 7 displays the error between all three models' predictions and labels, the actual stretch factors. The error is defined as the absolute value of the difference of the label and prediction



(a)



(b)

FIGURE 6: 100 POINT SAMPLE OF (a) TANH and (b) LOGARITHM MODEL OUTPUT OUTPUT VS LABELS

values

$$Error = |label - prediction| \tag{6}$$

since the stretch factors already serve as percent difference in velocity. It can be visibly seen in Figure 7 that scale transform has the worst results and as stretch factors reach extreme values, moving away from 1.0, the error gets progressively worse. That result is seen in all model errors but the worst with the scale transform. The log model error maintains low error even at extreme stretch factors. The scale transform has strange behavior around the extreme stretch factor of 0.6.

It has been proven that the standard feed-forward multilayer perceptron (MLP) with a single hidden layer can approximate any continuous function to any degree of accuracy [29]. However, to achieve desired levels of accuracy, network sizes increase dramatically with the complexity of the desired task. This explains why the machine learning models can accurately predict the stretch factor and why the tanh model does not perform as

Model	MSE
Scale Transform	0.0525
Standard machine learning Model	0.0084
Log machine learning Model	0.0037

TABLE 2: MSE CALCULATIONS FROM EACH MODEL

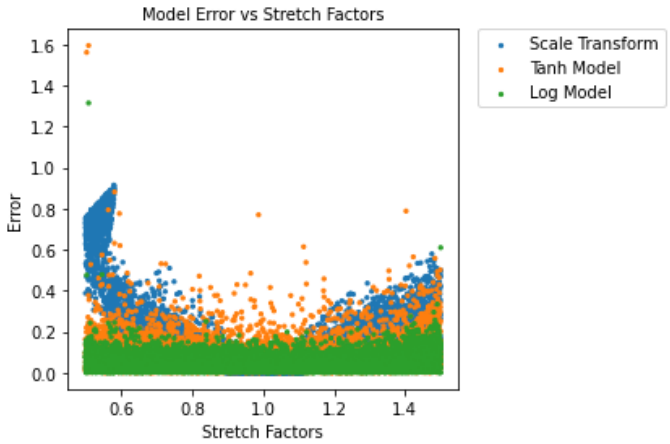


FIGURE 7: ERROR PLOT OF EACH MODEL

well as the log model. The problem at hand is complex, however, the machine learning models are able to learn the behavior.

It is believed that the natural logarithm model worked better due to the very nature of mathematical rules governing logarithms. The task involved in this research requires the computation of the Taylor series expansion of inner products. Transforming the non-linear function computation to the logarithm space changes multiplication to addition (4), this greatly eases the computation in learning, leading to a higher model performance [29]. Using e^x allows the model to exit the logarithm space after successful model parameter assignments [29]. Using logarithm forces an answer into the neural network [29]. FFNNs are not well suited to some non-linear approximations, including multiplication. It has been shown that logarithmic neural network architectures are better suited to non-linear function approximations, outperforming the MLP [29]. This intuition aids to explain why the natural log model outperforms the tanh model.

5 CONCLUSION

In order to achieve accurate damage detection in NDE, the full operation pipeline must be complete and without faults. This paper discussed the limitations in accuracy due to large variations in signal velocity in guided wave ultrasonics, causing temperature compensation techniques to fail. Our machine learning techniques, using a log-based FFNN, outperformed the prior art scale transform temperature compensation approach. With the success of the machine learning model in predicting stretch factors, it is likely that a better machine learning approach to temperature compensation exists.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under grant no. EECS-1839704.

REFERENCES

[1] Sohn, H., Farrar, C. R., Hemez, F., and Czarnecki, J., 2001. "A Review of structural health". *Library.Lanl.Gov*, pp. 1–7.

[2] Janapati, V., Kopsaftopoulos, F., Li, F., Lee, S. J., and Chang, F. K., 2016. "Damage detection sensitivity characterization of acousto-ultrasound-based structural health monitoring techniques". *Structural Health Monitoring,* ***15***(2), 2, pp. 143–161.

[3] Liu, C., Harley, J. B., Bergés, M., Greve, D. W., and Oppenheim, I. J., 2015. "Robust ultrasonic damage detection under complex environmental conditions using singular value decomposition". *Ultrasonics,* ***58***, 4, pp. 75–86.

[4] Rose, J. L., 2004. "Ultrasonic guided waves in structural health monitoring". *Key Engineering Materials,* ***270-273***(I), pp. 14–21.

[5] Harley, J. B., and Moura, J. M., 2012. "Scale transform signal processing for optimal ultrasonic temperature compensation". *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* ***59***(10), pp. 2226–2236.

[6] Abbas, M., and Shafiee, M., 2018. "Structural health monitoring (SHM) and determination of surface defects in large metallic structures using ultrasonic guided waves". *Sensors (Switzerland),* ***18***(11), 11, p. 3958.

[7] Zhu, X., Rizzo, P., Marzani, A., and Bruck, J., 2010. "Ultrasonic guided waves for nondestructive evaluation/structural health monitoring of trusses". *Measurement Science and Technology,* ***21***(4), 3, p. 045701.

[8] Roy, S., Lonkar, K., Janapati, V., and Chang, F. K., 2014. "A novel physics-based temperature compensation model for structural health monitoring using ultrasonic guided waves". *Structural Health Monitoring,* ***13***(3), 3, pp. 321–342.

[9] Croxford, A. J., Moll, J., Wilcox, P. D., and Michaels, J. E., 2010. "Efficient temperature compensation strategies for guided wave structural health monitoring". *Ultrasonics,* ***50***(4-5), 4, pp. 517–528.

[10] Lu, Z., Lee, S. J., Michaels, J. E., and Michaels, T. E., 2010. "On the optimization of temperature compensation for guided wave structural health monitoring". *AIP Conference Proceedings,* ***1211***(1), 3, pp. 1860–1867.

[11] Wang, G., Wang, Y., Sun, H., Miao, B., and Wang, Y.,

2019. "A reference matching-based temperature compensation method for ultrasonic guided wave signals". *Sensors (Switzerland),* ***19***(23), 11, p. 5174.

[12] Douglass, A. C., and Harley, J. B., 2020. "Model-based statistical guided wave damage detection for an aluminum plate:". *Structural Health Monitoring,* ***19***(6), 3, pp. 1937–1950.

[13] Harley, J. B., and Sparkman, D., 2019. "Machine learning and NDE: Past, present, and future". *AIP Conference Proceedings,* ***2102***(1), 5, p. 090001.

[14] Wunderlich, C., Tschöpe, C., and Duckhorn, F., 2018. "Advanced methods in NDE using machine learning approaches". *AIP Conference Proceedings,* ***1949***(1), 4, p. 020022.

[15] Melville, J., Alguri, K. S., Deemer, C., and Harley, J. B., 2018. "Structural damage detection using deep learning of ultrasonic guided waves". *AIP Conference Proceedings,* ***1949***(1), 4, p. 230004.

[16] Santos, A., Figueiredo, E., Silva, M. F., Sales, C. S., and Costa, J. C., 2016. "Machine learning algorithms for damage detection: Kernel-based approaches". *Journal of Sound and Vibration,* ***363***, 2, pp. 584–599.

[17] Svozil, D., Kvasnička, V., and Pospíchal, J., 1997. "Introduction to multi-layer feed-forward neural networks". *Chemometrics and Intelligent Laboratory Systems,* ***39***(1), 11, pp. 43–62.

[18] Jordan, M. I., and Mitchell, T. M., 2015. "Machine learning: Trends, perspectives, and prospects". *Science,* ***349***(6245), 7, pp. 255–260.

[19] Günther, F., and Fritsch, S., 2010. "neuralnet: Training of Neural Networks". *The R Journal,* ***2***(1), pp. 30–38.

[20] Ruder, S., 2016. "An overview of gradient descent optimization algorithms". *CoRR,* ***abs/1609.0***, 9.

[21] Bottou, L., 2010. "Large-scale machine learning with stochastic gradient descent". *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, pp. 177–186.

[22] Hecht-Nielsen, R., 1992. *Theory of the Backpropagation Neural Network*. Academic Press, 1.

[23] Sarle, W. S., 1994. "Neural Networks and Statistical Models". In Proceedings of the Nineteenth Annual SAS Users Group International Conference, SAS Institute Inc., pp. 1538–1550.

[24] Bishop, C. M., 1994. "Novelty detection and neural network validation". *IEE Proceedings: Vision, Image and Signal Processing,* ***141***(4), 8, pp. 217–222.

[25] Goldberg, X., 2009. "Introduction to semi-supervised learning". *Synthesis Lectures on Artificial Intelligence and Machine Learning,* ***6***, 6, pp. 1–116.

[26] Su, Z., Ye, L., and Lu, Y., 2006. "Guided Lamb waves for identification of damage in composite structures: A review". *Journal of Sound and Vibration,* ***295***(3-5), 8, pp. 753–780.

[27] Ritchie Ng, J. F., 2019. "Deep Learning Wizard". *https://zenodo.org/record/2644957*, 4.

[28] Sibi, P., Allwyn Jones, S., and Siddarth, P., 2013. "Analysis of different activation functions using back propagation neural networks". *Journal of Theoretical and Applied Information Technology,* ***47***(3), pp. 1344–1348.

[29] Hines, J. W., 1996. "A Logarithmic Neural Network Architecture For Pra Approximation". *Proceedings of the 1996 American Nuclear Society, International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies,* ***1***, 2, pp. 235–241.

[30] Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A., 2018. "How does batch normalization help optimization?". *Advances in Neural Information Processing Systems,* ***2018-Decem***, 5, pp. 2483–2493.

[31] Wallach, D., and Goffinet, B., 1989. "Mean squared error of prediction as a criterion for evaluating and comparing system models". *Ecological Modelling,* ***44***(3-4), 1, pp. 299–306.

[32] Feurer, M., and Hutter, F., 2019. *Hyperparameter Optimization*. Springer International Publishing.