Tech Science Press

Applying Neural Networks for Tire Pressure Monitoring Systems

Alex Kost[1], Wael A. Altabey[2,3,4], Mohammad Noori[1,2,*] and Taher Awad[4]

[1]Mechanical Engineering Department, California Polytechnic State University, San Luis Obispo, CA 93405, USA.
[2]International Institute for Urban Systems Engineering (IIUSE), Southeast University, Nanjing, China.
[3]Nanjing Zhixing Information Technology Co., Ltd., Nanjing, China.
[4]Mechanical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, 21544, Egypt.
[*]Corresponding Author: Mohammad Noori. Email: mnoori@outlook.com.

Abstract: A proof-of-concept indirect tire-pressure monitoring system is developed using artificial neural networks to identify the tire pressure of a vehicle tire. A quarter-car model was developed with MATLAB and Simulink to generate simulated accelerometer output data. Simulation data are used to train and evaluate a recurrent neural network with long short-term memory blocks (RNN-LSTM) and a convolutional neural network (CNN) developed in Python with Tensorflow. Bayesian Optimization via SigOpt was used to optimize training and model parameters. The predictive accuracy and training speed of the two models with various parameters are compared. Finally, future work and improvements are discussed.

Keywords: RNN-LSTM; CNN; artificial neural networks; tire pressure monitoring systems

1 Introduction

It is difficult to understate how important properly pressurized tires are to the performance and safety of a vehicle and its operator, respectively. The National Highway Traffic Safety Administration (NHTSA) estimates that 11,000 tire-related crashes occur annually in the US, with 200 people estimated to be killed in these crashes [1]. Furthermore, under-inflated tires contribute to the following performance issues when driving [2]:

(1) Poor fuel economy, wasting an estimated 3.5 million gallons daily and costing drivers as much as 11 cents per gallon in the US.

(2) Longer stopping distances and sluggish/ineffective handling, resulting in more dangerous driving conditions.

(3) Faster tire wear, reducing the average life of a tire by 4,700 miles.

Tire-pressure monitoring systems (TPMS) became federally mandated in 2000 by the Transportation Recall Enhancement, Accountability, and Documentation Act, where legislators ruled to require a warning system in new motor vehicles to indicate to the operator when a tire is significantly under inflated [3]. More specifically, all motor vehicles must have a system that is capable of detecting when one or more of the vehicle's tires, up to all four tires, is 25% or more below the manufacturer's recommended inflation pressure or a minimum activation pressure specified in the standard, whichever is higher [4]. Nonetheless, a study performed in April 2009 showed that 45% of TPMS-enabled vehicles still have under-inflated tires [5]. Therefore, for obvious moral and legal reasons, it is imperative that drivers know that their tires are inflated properly. It is in the individual's and society's best interests to improve safety, performance, and savings while on the road.

Extensive work has also been reported in the literature pertaining to the TPMS. For instance, [6] developing a new algorithm based on extreme value statistics. Gao et al. [7] conducted a variety of research on TPMS standards and test methods. He also designed the special test device. The Direct monitoring systems is depend on integrated sensors. Vibration signals usually present different parameters, including velocity, displacement or acceleration that can be measured by a velocity sensor, a displacement probe or an accelerometer, respectively [8-12]. Persson et al. [13] presented an indirect TPMS system using sensor fusion. He introduced the yaw rate to fix the wheel rolling radius. Luo [14] used the resonant frequency of tires to monitor tire pressure and estimated resonant frequency by analyzing the frequency spectrum of wheel speed. For wheel speed, when the tire pressure is insufficient, the rolling radius decreases and the rotational speed increases [15]. Han et al. [16] studied methods of monitoring tire pressure. of the vehicle when it is in a straight line and turns. The turning radius can be obtained depending on the vehicle geometry parameters [17]. Changzheng et al. [18] developed a novel surface-micromachining technology to monolithically integrate piezoresistive pressure sensor and accelerometer for tire-pressure monitor system (TPMS) applications. Daniel et al. [19] derived a real-time physical model for strain-based intelligent tires has been. They provide circumferential strains of the tire inner and conducted experiments on a strain-based intelligent tire. Yu-Jen et al. [20] develop and analyzed a nonlinear suspended energy harvester (NSEH) that can be mounted on a rotating wheel. Oche et al. [21] proposed an innovative decision rule-based approach to tyre monitoring. This approach relies on the Dominance-based Rough Set Approach (DRSA), which is a well-known multicriteria classification and preference learning method. Raul et al. [22] developed and tested In-wheel sensor system for pneumatic tires. The System tested using a drum-test machine over a wide range of conditions. They developed signal processing methods and measures of tire-terrain contact.

The main purpose of engineering structural design is to meet the functional requirements of the system in the most economical way, and the reliability is the effective control method to meet this purpose. Reliability theory began in the 1940s. The earliest requirement to use the reliability is the military needs to improve the reliability of electronic components [23,24]. Menglong, and Dongyuan [25] proposed a method based on reliability and falsity in order to solve the errors of D-S evidence theory when there is inconsistent and conflicting among the evidences. Firstly, they calculated the reliability of each evidence in the system identification framework according to the Lance distance, then, they adopted the falsity of evidence to measure the degree of conflict between different evidences, combining reliability and falsity determines corrected coefficient of evidences, and basic probability assignment is reconfigured, finally they modified the basic probability assignment are fused by D-S combination rule. Binwen [26] designed Direct-type tire pressure monitoring module based on sensors, and also he designed the central receiving module using microcontroller. Thus, the real-time monitoring of tire pressure and temperature is achieved. Cullen et al. [27] employed the new system useing the CAN bus network technique, as well as a novel method of relaying the tire pressure status off the wheel without using any power or transmitter system, thus overcoming many of the obstacles faced by systems of the same scope. Jingui et al. [28] developed On-vehicle triboelectric nanogenerator (V-TENG) as a direct power source for tire pressure monitoring system. They achieved the high performance of the V-TENG with wide ranges of temperature, rotation speed and magnetic force, and improved the durability and reliability of V-TENG for long-term operation. Hongjip et al. [29] proposed an energy harvester for rotating systems under modulated noise excitations by taking advantage of self-tuning stochastic resonance with particular application to power smart tires, that compared to existing tire energy harvesters, it has larger power output and wider bandwidth. They conducted the Numerical simulation to simulate the harvested power in a passenger car tire at different driving speeds. To validate the simulation results, thy conducted the experiment model, they show that the experiment results show good agreement with the numerical simulation, which proves the feasibility of the proposed harvester.

The most commonly used TPMS in vehicles today is a simple pressure sensor mounted within the tire to directly measure the pressure of the air within the tire. When the integrated battery dies on these sensors, the sensors must be replaced manually.

Time, money, and labor are spent to replace this simple sensor. It would be advantageous if the TPMS architecture was created such that maintenance and repair were not needed.

As advancements in machine learning and deep learning techniques continue, it is no longer a question of how or why to apply these techniques, but where to apply them.

In this work, a proof-of-concept TPMS architecture is suggested that uses accelerometer data and an artificial neural network (ANN) to determine whether the tires on a vehicle are under, over, or nominally inflated.

2 Background

Background research for this work focused on three fields: representations of suspension systems, current TPMS architectures, and a high-level overview of ANNs.

2.1 Suspension Representation

A simplified Quarter Car Model representation of a vehicle suspension system is used in this work. The representation only models vertical movement (1 degree of freedom) and assumes that the vehicle is rigid; only vibrations transferred from the ground to the tires, axles, and suspension systems are considered. This representation also does not consider any forces or reactions due to the geometry of the vehicle; it is only looking at a single wheel on this vehicle. The representation is presented in Fig. 1 [30].

The analytical model utilized in this work is very simple, purposefully. The focus of this research is on the development of an intelligent algorithm for tire pressure measurement rather than creating and using a complex dynamic model. There is much work that already exists to properly model passenger cars for dynamic analysis. For instance, Tan and Wang [31] and Liu [32] works are two examples of more thorough rigid body analysis applied on passenger motorized vehicles. Where their works end with model validation, this work will purposefully use a simpler and more practical dynamic model to validate the algorithm rather than the model.

The unsprung mass mu refers to all masses that are attached to and not supported by the spring, such as the wheels, axles, or brakes. In this representation, the unsprung mass is the weight of the tire and the weight of the air of the tire. In an actual vehicle, suspension stiffness and damping values k_s and c_s are functions of suspension type, tire geometry, tire pressure, vehicle geometry, and vehicle weight. These values should be constant in vehicles without active suspension systems, so the only changing parameter in this model is the unsprung mass's stiffness k_u. Any damping in parallel with k_u is negligible with respect to cu and is thus not included in the representation.

2.2 TPMS Architectures

The NHTSA provides vehicle manufacturers three ways to comply with the law: direct, indirect, and hybrid TPMS [33]. Direct TPMS consists normally of pressure sensors located inside each wheel to directly measure the pressure in each tire. Indirect TPMS compares speed data collected from vehicle's anti-lock braking system wheel speed sensors to compare rotational speeds of tires against one another to determine the pressure. Direct systems are more accurate and precise, whereas indirect systems are less hardware-dependent and more robust for each vehicle. The NHTSA leaves the definition of a hybrid TPMS purposefully vague and suggests such a system would use a combination of direct and indirect methods to fulfill the regulatory requirements.

As described by Transport & Environment (T&E) [34] indirect TPMS is unable to accurately measure tire pressure in real-time. Required routine recalibration, requires the vehicle to be moving linearly to work, and can falsely trigger based on road conditions affecting wheel rotation and vibration. T&E asserts that indirect TPMS systems generally comply with regulatory requirements but "show very poor performance" when testing in more realistic conditions. For these reasons, direct TPMS is currently the more commonly-applied technology in vehicles today. However, their placement in tires requires time, money, and labor in case repairs or replacements must be made.

Research in indirect and hybrid TPMS architectures has grown and continues to grow because of their perceived advantages over direct TPMS as computing power increases. For example, Persson et al. [13] presented an indirect TPMS combining vibration and wheel radius analyses that was able to detect pressure losses larger than 15% in one, two, three, or four tires and identify the under-inflated tire within 1 minute. Wang et al. [35] improved the indirect TPMS algorithms with the inclusion of accurate pressure identification under steering conditions.

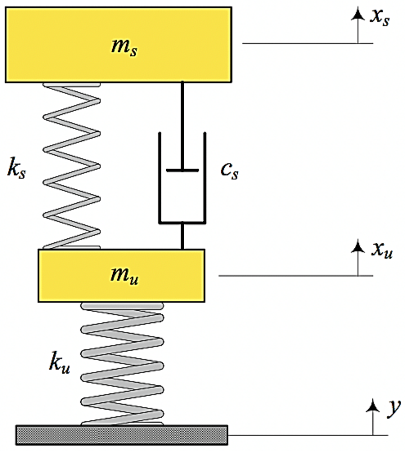


Figure 1: A free-body diagram of the quarter-car model [30]

2.3 Artificial Neural Networks

An artificial neural network (ANN) is a machine learning algorithm used to solve advanced non-linear problems such as handwriting or speech recognition [36-44]. Neural networks connect computational nodes together to form a singular network, where each computational node is performing a calculation on its input and outputting the result to all outgoing connections. The output of a node can be the input to at least one other node or too many other nodes. Outputs can be scaled and biased by weights and biases respectively; think the canonical linear function y = mx + b, where y is the original output, m is the weight, x is the new output, and b is the bias. Often, activation functions are added to the networks; these further define the output with a linear or non-linear function. As shown by Ramachandran et al. [45] the most commonly used activation function in deep learning projects is the rectified linear unit (ReLU). In summary, interconnected computational nodes perform linear and non-linear operations on inputs.

At first, all ANN models do not perform well because the weights and biases are not tuned; that is, the model is not trained. Neural networks can learn a hierarchical feature representation from raw data automatically [46]; that is, they \learn or can be trained through example. In this work, we train our models via supervised learning| that is, with labeled training data-and compare the model's predictions to the actual labels. By repeatedly minimizing the error between prediction and truth, the model updates the trainable parameters and its accuracy improves. This updating is based on minimizing a cost (generally inversely proportional to accuracy) via some optimization strategy. Gradient Descent strategies are often implemented; in this work, the Adaptive Moment Estimation (Adam) strategy is applied. Adam computes adaptive learning rates for each parameter and takes advantage of the idea of momentum to more quickly converge on the global minima with reduced oscillation [47].

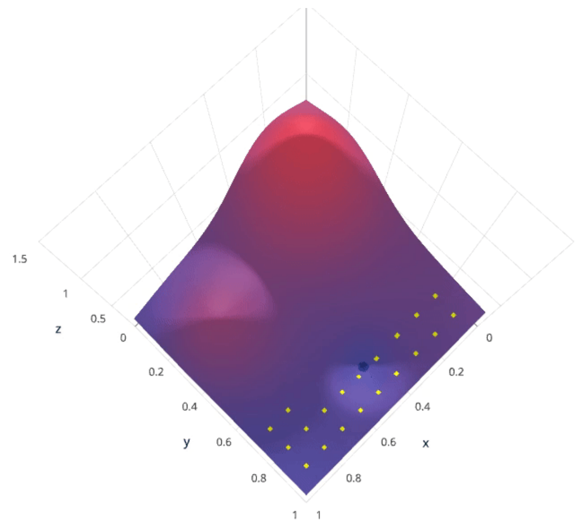
Bayesian optimization is a powerful tool for optimizing objective functions which are very costly or slow to evaluate [48-53]. In particular, we consider problems where the maximum is sought for an objective function *f*:

$$x_{opt} = \arg\max_{x\in\chi} f(x) \tag{1}$$

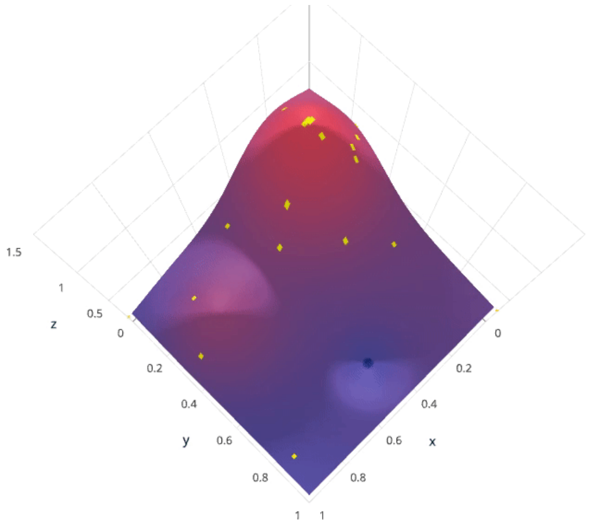
where χ is some design space of interest; in global optimization within a domain f: $\chi \subset \mathbb{R}^d$ which is a bounding box (tensor product of bounded and connected univariate domains). Numerous strategies for modeling f in the Bayesian optimization setting have been suggested, including the use of Gaussian processes [50,54], random forests [55]; and tree structured Parzen estimators [56,57].

Furthermore, models hyperparameters can be tuned such that they can more quickly be trained and perform more optimally. Grid search tuning is a standard method where an exponentially large grid of possible hyperparameter combinations is systematically searched. Alternatively, Bayesian Optimization tuning promises a more intelligently search by learning from prior hyperparameter combinations and their results to intelligently suggest better combinations [58]. Grid searches are exponentially expensive whereas Bayesian optimization are only linearly expensive, as visualized in Fig. 2. In this work, the software-as-a-service product SigOpt is applied to perform Bayesian optimization techniques for quick, intelligent tuning.

The type of input data generally defines the type of ANN to be used; in this case, the models are interpreting time series data. As defined by Georg [59], a time series is a sequence of vectors depending on time t such that; and so on. The components of at each time t (referred to as datapoints in this work) are distinct from one another but are not informative enough to extrapolate meaningful information from the time series; instead, each datapoint in a time series must be analyzed in relation to the rest of the time series.



(a)



(b)

Figure 2: (a) Grid Search vs. (b) Bayesian Optimization techniques for tuning, where each yellow dot indicates a model evaluation. Notice that grid searches could be searching along a potentially-coarse grid, whereas Bayesian optimization techniques test any possible combination within the space and intelligently suggests combinations to reach optimal solutions with fewer evaluations

We discuss two major model types for interpreting time series data in Fig. 3 the recurrent neural network (RNN) and convolutional neural network (CNN).

Convolutional neural networks (CNNs) interpret clusters of datapoints (e.g., time-series, images, sentences, sound recordings, so on) together to preserve spatial or temporal relationships. CNNs apply kernels or filter, i.e., a weight matrice to recognize and extract features or patterns [60-63].

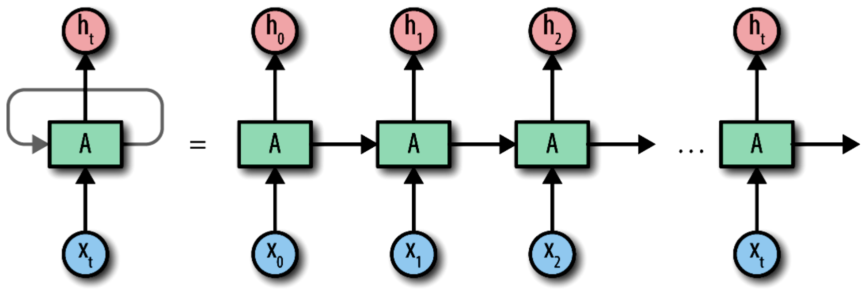


Figure 3: A visual representation of a single block in a recurrent neural network (RNN). Taken from Olah [64]

Recurrent neural networks (RNNs) interpret time-series data successfully by adding feedback loops to the standard ANN network architecture [65,66]. Some RNNs use more complex computational nodes known as long short-term memory (LSTM) blocks to mitigate an issue common in RNNs known as the vanishing gradient problem in Fig. 4 [67].

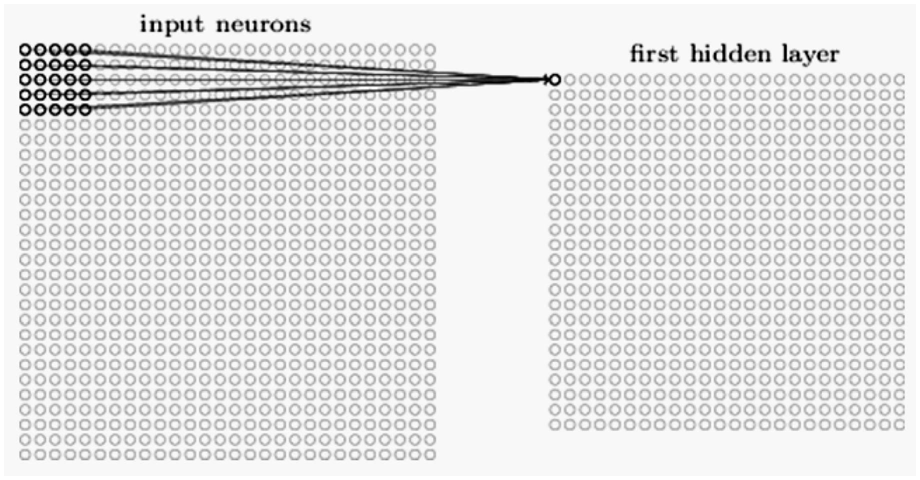


Figure 4: Visualization of a 5 × 5 filter convolving around an input volume and producing an output. Taken from Adit [67]

The first few layers of a typical ANN act as feature extractors; that is, they are responsible for extracting meaningful information from the input data. For example, RNNs build an internal memory and CNNs use pattern matching. This meaningful information is then fed into a classifier. Classifiers are generally fully-connected layers (each node is connected to one another; see Fig. 5 with n outputs, where n is the number of classes in the input data.

ANNs have been applied in the automotive industry for decades. In 1990, Wiggins et al. [68] presented a neural network that could identify engine faults based on the vehicle's engine controller data. Neural networks were used to control the air-to-fuel ratio in fuel injection systems as shown by Alippi et al. [69]. More recently, ANNs have driven advances in automated vehicle control (\self-driving) that can detect, identify, and respond to objects and pedestrians on the road in real time.

While Tesla, Mercedes-Benz, and BMW were first introduce these features to consumer vehicles, the technology is becoming increasingly ubiquitous [70]. A NHTSA investigation conducted in January 2017 found crash rates Tesla crash rates have dropped by almost 40% since enabling self-driving capabilities in 2015 [71].

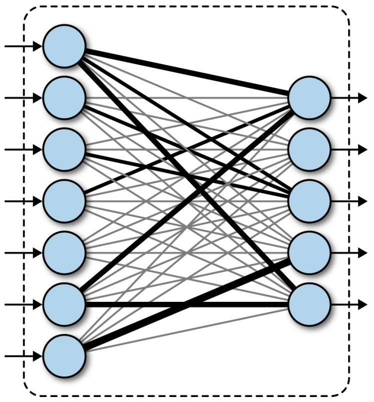


Figure 5: Visualization of a fully-connected layer. Taken from Hollemans [72]

3 Research Work

With the desire to explore alternative indirect TPMS frameworks and inspired by deep learning is seemingly infinite applications, this work explores a deep learning framework that analyzes vehicle suspension acceleration data to classify the vehicle tires as under-inflated, nominally inflated, or over-inflated. To validate this idea, work was broken into the following sections:

(1) Collecting Data. The accuracy and capability of the ANN is largely dependent on the size of our data-ANNs tend to improve when there is more data for training. In this work, data was simulated by a quarter-car model written in MATLAB and Simulink. The data serves as the training, validation, and test sets for the ANN.

(2) Creating the Algorithm. Using the data from the prior step, an RNN-LSTM and CNN are developed in Python with Google's open-source TensorFlow API. Tuning model and training parameters are done using Bayesian Optimization via SigOpt (All source code is available on Olah [64]).

3.1 Collecting Data

A MATLAB model for the quarter-car representation as shown in Fig. 1 was run at various tire pressures and step-sizes to generate simulated examples of a vehicle suspension system experiencing a step response (in an attempt to be analogous to a pothole or speed bump). The simulation solves the system of ordinary differential equations for every time step for the position, velocity, and accelerations of the sprung mass m_s and unsprung mass mu. The simulation inputs are presented below in Tab. 1 and their accompanying derivations are presented in Appendix A.

Table 1: Simulation input variables

Variable	Description	Value	[Units]
p_u	Tire pressure	Varies	[psi]
y	Step size	Varies	[m]
m_s	Sprung mass	277.25	[kg]
m_u	Unsprung mass	34.69	[kg]
k_s	Sprung stiffness	557.97	[N/m]
c_s	Sprung damping	6218.35	[N-sec/m]
k_u	Unsprung stiffness	Varies	[kPa]
g	Gravity	9.81	[m/sec^2]

The simulation was performed for pu = 25.5, 26, 26.5, 27,…, 38.5 and for y = 0.10, 0.15, 0.2,..., 2.0, generating 633 total examples. Every 1.5-second-long run is composed of 1500 data points and labeled according to the inflation classifications as defined by Tab. 2.

Table 2: Inflation classifications, pressures, and labels

Inflation clarification	Pressure Range (psi)	Label (int)
Under	26-30	0
Nominal	30-34	1
Over	34-38	2

These classifications are 10% of 32 psi, well within the 25% specification as defined by the TREAD Act. The label of the simulation and the sprung's mass acceleration are saved in individual .csv files to be parsed by the algorithm. An example of the generated data is presented below in Tab. 3 (note that the first row is only shown here for clarification and is not included in the raw output).

Table 3: Example of simulated data: Sim_35.5psi_0.75m.csv

Label	$\ddot{x}_s$, t = 0:000 s	$\ddot{x}_s$, t = 0:001 s	.....	$\ddot{x}_s$, t = 0:420 s	$\ddot{x}_s$, t = 0:421 s
2	-0.00073852	-0.00067152	….	-1.3974	-1.2822

Plots were developed of x_s vs. time as a quick sanity check is shown in Fig. 6. The plots make intuitive sense-higher pressure correlates with greater stiffness, which then increases the natural frequency, slows the settling speed of the mass, and reduces the maximum amplitude. The simulation is sound.

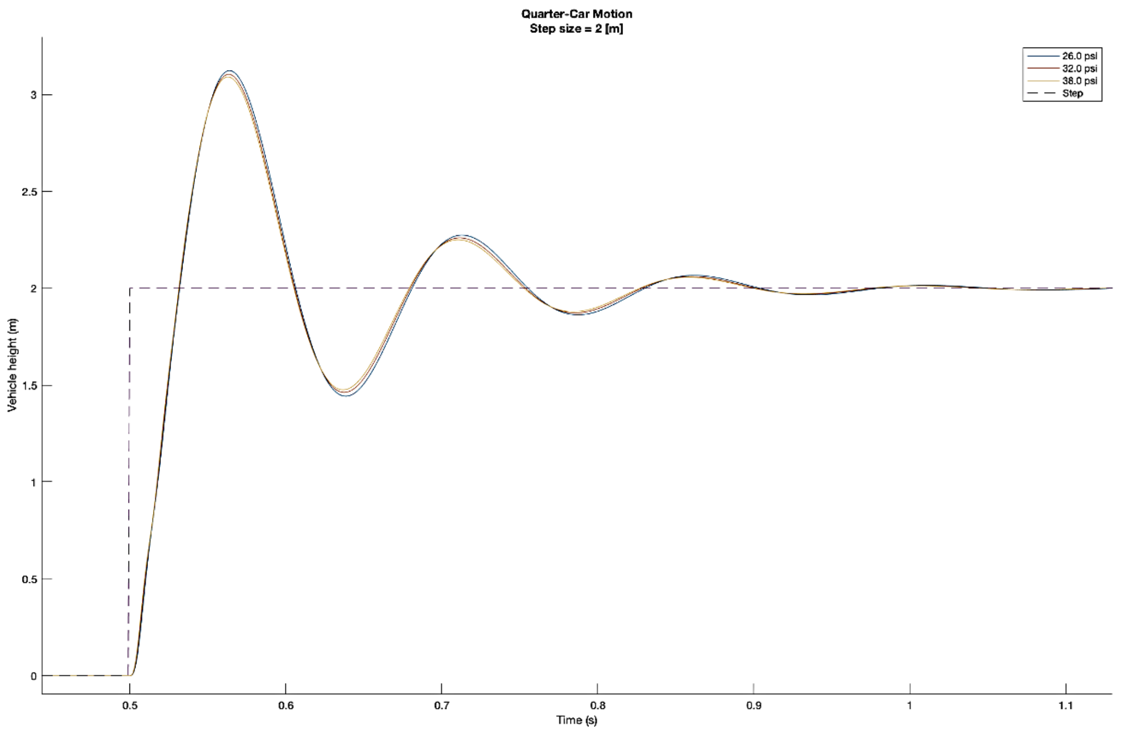


Figure 6: Sprung mass *vs.* time for p_u = 26, 32, 38 psi for step size y = 2 m

3.2 Building the Algorithm

TensorFlow was used to build, train, and evaluate a RNN and a CNN. Development of each neural network followed the same specifications as listed below.

(1) Import the simulated data into the Python environment.

a. The input data shall be shuffled randomly.

b. The input data shall be split into a training set (60%), validation set (20%), and test set (20%). Use of a validation and test set reduced the chance of overfitting and follows the commonly-used 3:1:1 ratio suggested by machine learning experts [73].

(2) Build the model of the neural network.

a. The model shall be fed labeled input data and output predicted labels.

b. The input data should be fed in batches to minimize computational load between parameter updates. Generally, the recommended starting batch size is 32 [74].

c. The model shall prevent overfitting by applying dropout to the outputs of at least one fully-connected layer [75].

d. Batch normalization shall be applied after various layers to reduce the internal covariate shift within the model [76].

e. Model logits shall be converted to classification predictions using the softmax activation function.

(3) Evaluate the predictive capabilities and training speed of the model.

a. The cost shall be calculated using the cross-entropy function between the input data labels and model predictions [77].

b. The accuracy shall be calculated by comparing the model's predicted labels to the input data labels.

c. The training speed shall be minimized by tuning the model hyperparameters.

(4) Train the model parameters.

a. The training shall end after a predefined number of epochs and not be stopped early to observe any overfitting in the model.

b. The training method shall minimize the batch's average cross-entropy loss using Adam Optimization strategy [47].

c. The learning rate shall be static or exponentially decaying.

3.3 Development

The RNN-LSTM and CNN models are self-contained in RNNModel and CNNModel respectively as shown in Fig. 7. Both models are similar except for the feature extraction near the input layer of the model.

A DataProcessor class was written to provide methods to scan a directory for all files and perform various preprocessing operations. In this work, DataProcessor scans the simulated data directory; generates lists of all files found across all labels; shuffles and splits the filenames across test, validation, and training sets; and loads the feature data and label data found in each files from each set into member variables to be used for training. The training class TrainModel is the entry point to train the model. Instantiating TrainModel builds the desired model with a provided learning rate learning rate and dropout rate dropout rate. Calling train model trains the model for a desired number of epochs n epochs using feature and label data inherited from DataProcessor. Every $\frac{1}{n\_checks}$, the model's accuracy and cost are evaluated across the entire training and validation datasets and reported to TensorBoard for visualization. The test set accuracy is evaluated before and after training.

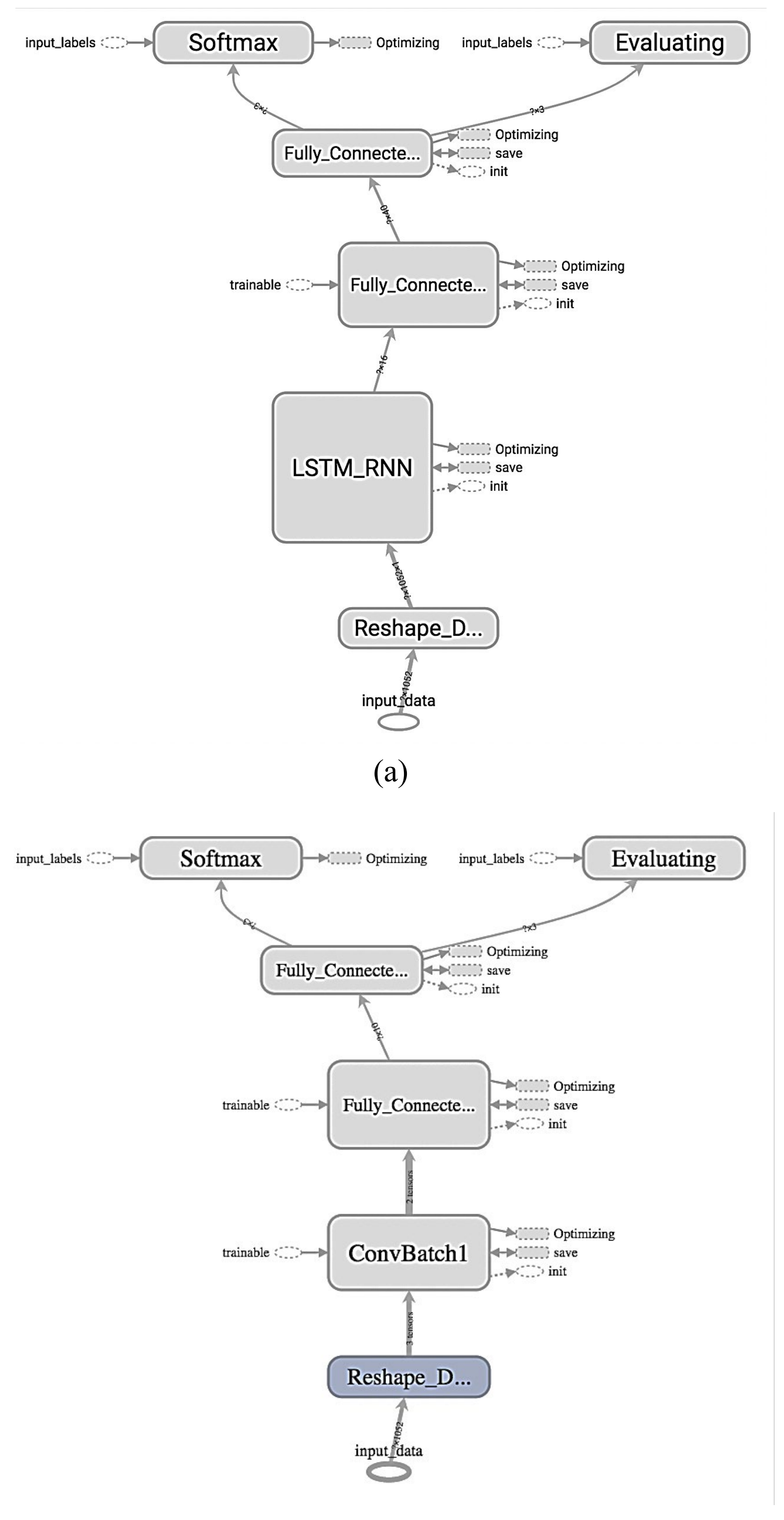


Figure 7: (a) RNN-LSTM and (b) CNN model visual graphs as created by TensorBoard

3.4 Tuning

The model parameters were tuned via SigOpt to identify optimal values for various model hyperparameters. Tuning classes GridSearchTune and SigOptTune were developed to perform a grid search or connect to SigOpt to perform a Bayesian search respectively. It was estimated that a grid search over the entire model space would take over two weeks of computations per model, whereas SigOpt's more-intelligent Bayesian search strategy would take days instead. Thus, only SigOptTune was used in this work.

Two SigOpt experiments were run for each model to optimize the training speed and accuracy respectively. The parameters under investigation are listed below in Tab. 4.

Table 4: Parameters optimized via SigOpt Bayesian optimization. *denotes that the parameter is related to Adam optimization strategy

Name	Description	RNN-LSTM	CNN
dropout rate	Dropout rate	X	X
*learning rate**	Learning rate	X	X
*beta1**	1st moment estimates exponential decay rate	X	X
*beta2**	2nd moment estimates exponential decay rate	X	X
*epsilon**	Numerical stability constant	X	X
Num_filt _1	Number of filters in convolutional layer		X
Kernel_size	Kernel size in convolutional layer		X
Num_fc_1	Number of neurons in first fully-connected layer	X	X
n_layers	Number of hidden layers in model	X	
n_hidden	Number of features per hidden layer in LSTM	X	

4 Results and Discussion

4.1 Initial Results

Tuning the Adam-specific hyperparameters gave insight in a recurring issue with the LSTM-RNN: The model would not improve in performance after 200 steps (40 epochs with batch size = 128). Fig. 8 shows multiple training curves with various values for learning_rate, beta _1, beta_2, and epsilon. Where the cross-validation accuracy would remain at 33.3%, or the same accuracy as randomly guessing.

These results can be from the RNN-LSTM's inability to identify any meaningful features after 40 epochs of the 633 training examples. The same results were seen when the model was trained for 200 epochs: The RNN-LSTM underfit the simulated data every time. Therefore, all model hyperparameters were increased. The resulting models successfully fit the input data and achieved significantly better accuracy when classifying the test set data. Further hyperparameter tuning showed that increasing the number of layers to be greater than 1 result in the model fitting the data appropriately. After 100 observations, Sigopt reported the RNN achieved 96.2% accuracy.

The CNN did not require much hyperparameter tuning. The CNN achieved near state-of-the-art success (accuracy > 95%) on the first try. The CNN achieved 100% accuracy after 15 optimization evaluations with SigOpt.

The final model hyperparameters were based on the first evaluation that classified the test set with 100% accuracy. These values are shown in Tab. 5. Similarly, the final performances are shown below in Fig. 9.

Table 5: Final hyperparameters chosen for both models

Name	RNN-LSTM	CNN
dropout rate	0.672	0.309
*learning rate**	0.00001	0.033
*beta1**	0.9	0.684
*beta2**	0.999	0.845
*epsilon**	1e-08	0.282
Num_filt _1	-	16
Kernel_size	-	4
Num_fc_1	31	6
n_layers	4	-
n_hidden	22	-

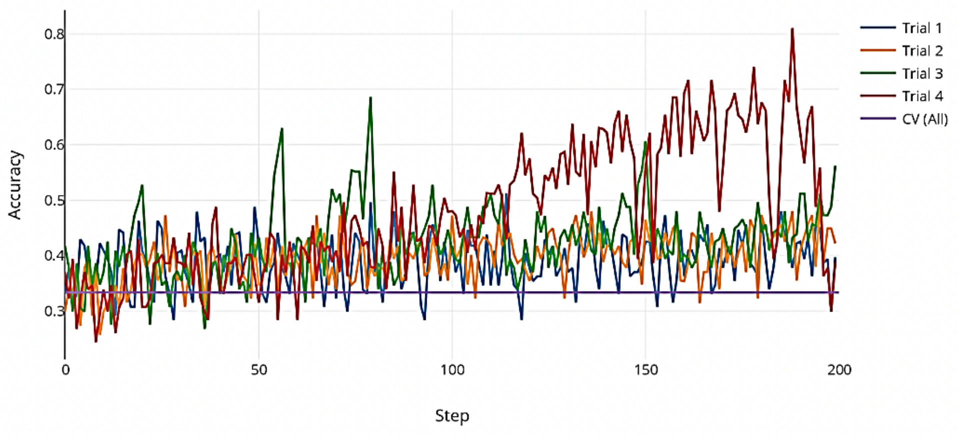


Figure 8: RNN-LSTM: Training classification accuracy for various Adam optimization strategy optimization parameters learning_rate, beta _1, beta_2, and epsilon

4.2 Final Results and Discussion

Overall, both CNN and RNN models achieved above 90% accuracy on the validation and test dataset given sufficient time. Fig. 9 depicts the accuracy curves during training across the training and validation datasets.

Different training parameters and hyperparameters were defined for each model to achieve these results. The training parameters of both models saw a change in the batch size batch size and number of epochs n_epochs. The batch size was increased from 32 to 256 so each model update would better represent the dataset. The models were ran until a validation dataset accuracy above 90% was observed, hence the final value n_epochs = 1000.

The CNN requires significantly less time to train than the RNN-LSTM. This can be explained by looking at the mathematics behind the architectures. At each time step t, a RNN-LSTM must perform the following computations:

$$\begin{cases} g^u = \sigma\left(W^u h_{t-1} + I^u x_t + b_u\right) \\ g^f = \sigma\left(W^f h_{t-1} + I^f x_t + b_f\right) \\ g^o = \sigma\left(W^o h_{t-1} + I^o x_t + b_o\right) \\ g^c = \tanh\left(W^c h_{t-1} + I^c x_t + b_c\right) \\ m_t = g^f \Theta m_{t-1} + g^u \Theta g^c \\ h_t = \tanh\left(g^o \Theta m_t\right) \end{cases} \tag{2}$$

where σ is the logistic sigmoid function, Θ represents elementwise multiplication, W^u, W^f, W^o, W^c are recurrent weight matrices, I^u, I^f, I^o, I^c are projection matrices, b is the bias vector, and h and m are hidden and memory vectors responsible for controlling state updates and outputs [78]. On the other hand, the input to some unit x_i^l in layer l is the sum of the previous layer's cells contributions y multiplied by a filter ω with size m [79]. More clearly,

$$x_i^l = \sum_{a=0}^{m} \omega_a y_{i-a}^{l-1} b_i \tag{3}$$

Compared directly against the fundamental equations behind a 1D convolution layer, one can see a stark contrast in complexity. Even if the filter or the number of previous-layer inputs are large in size, the CNN model is significantly simpler than the RNN-LSTM model and thus is easier and faster to train.

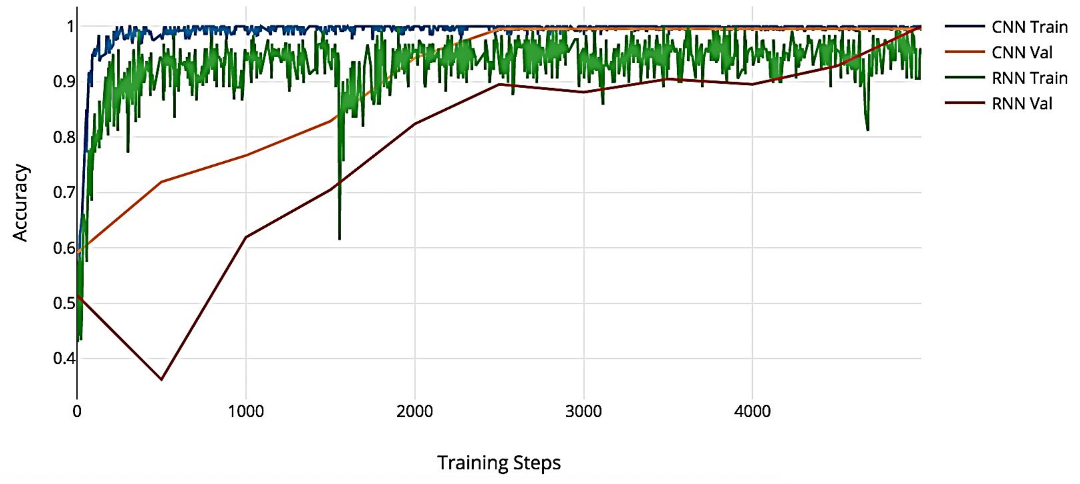


Figure 9: Classification Accuracy during Training

The CNN also outperformed the RNN-LSTM model in classification capability. The RNN-LSTM model feeds the hidden layer from the previous layer from the previous step into the next step to provide information for tasks requiring long-range contextual information, but the input data here is based on short, simulated step responses. The additional computations are not needed for classifying the data in this work; in fact, the RNN-LSTM is incorrectly biased on the built-up memory. The CNN is looking for specific patterns within windows of time within the time-series data. The clean, short simulated data does not vary in sequence length and has repeatable patterns within the data so the CNN is able to quickly train and accurately classify input data.

Disadvantages of the CNN in the application of tire-pressure monitoring are the same as any time-based series applications. The performance of CNN is dependent on the quality and size of the dataset; in this work, the dataset is small and clean. The patterns within each time series are occurring at the same time across all of the samples. CNNs also generally suffer from overfitting and poor random initialization issues, but in this work, neither issue was observed. More work must be done using a more complex dataset to further understand the weaknesses of the CNN.

4.3 Future Work

This work laid down a foundation to explore an ANN-based TPMS, but much more work needs to be done before this technology can be applied. Future work should attempt to address the following aspects not covered here:

(1) Improve the simulated data. In this work, all data was generated from a quarter-car model simulation. Many assumptions were considered in the simulation and thus, the simulation presented herein is not a good representative of a real car model. A better simulation can be made by using a half-car or full-car model instead of a quarter-car model or considering more degrees of freedom. Tan or Liu's research for more-accurate vehicle modeling may be a useful starting point.

(2) Collect experimental data. Even better than simulated data is real experimental data. Collecting and analyzing real data can result in a better, more generalized classifier with no issues arising from training on simulated data. Furthermore, the data should be generalized away from a step function profile to the acceleration profile of general driving such that the TPMS can identify under pressurized tires at all times.

(3) Develop the hardware. Instead of assuming the computational and electrical power required for the system exists, a more-thorough investigation should be performed to determine the validity of the claim. A theoretical system with the properly specified requirements would bring this work one step closer to reality.

(4) Improve the algorithm. Further fine-tuning of the training parameters and hyperparameters as well as adding and removing layers and features from the model architecture may result in more efficient and effective models. Fawaz et al. [80] comparison of deep learning models for time series classification may be a valuable starting point for identifying better models, especially with regard to improving the CNN architecture performance.

5 Conclusion

Considering the various limitations of the work, an ANN-based TPMS is far away from being applied across the automotive industry. Nonetheless, this work showed that both a CNN and RNN-LSTM model can be developed and trained on simulated training data to accurately classify unseen simulation data. This proves the algorithm's ability to identify unique patterns across each class and sort accordingly, all without any explicit instruction on the mechanical principles behind the data. With better data and appropriate hardware, vehicles may one day be equipped with ANN-based TPMS.

Acknowledgement: This research project was made possible in part due to a release time funding from Donald E. Bently Center for Engineering Innovation, Mechanical Engineering, Cal Poly. Herein this support is acknowledged.

References

1. NHTSA (2016). Tires URL. https://www.nhtsa.gov/equipment/tires.
2. Tire Wise (2016). Tire Maintenance URL. https://www.safercar.gov/tires/pages/tires_maintenance.
3. United States Senate and House of Representatives (2000). Transportation Recall Enhancement, Accountability, and Documentation (TREAD) Act. https://www.congress.gov/106/plaws/publ414/PLAW-106publ414.pdf.
4. NHTSA (2005). Docket No. NHTSA 2205-20586.
https://one.nhtsa.gov/cars/rules/rulings/tpmsfinalrule.6/tpmsfinalrule.6.html.

5. NHTSA (2009). Tire Pressure Maintenance-A Statistical Investigation. https://www.congress.gov/106/plaws/publ414/PLAW-106publ414.pdf.

6. Kan, Y. (2015). *Research on algorithm of indirect TPMS based on statistics of wheel speed extremum.* Yanshan University, Qinhuangdao.

7. Gao, M., Xu, Z., Zhao, B. (2008). Analysis and study on test methods of tire pressure monitoring system. *Automotive Technology, 44-47.*

8. Zhao, Y., Noori, M., Altabey, W. A., Zhishen, W. (2018a). Fatigue Damage Identification for Composite Pipeline Systems Using Electrical Capacitance Sensors. *Smart Materials and Structures, 27 (8),* 0850.

9. Altabey, W. A., Noori, M. (2017a). Detection of fatigue crack in basalt FRP laminate composite pipe using electrical potential change method. *Journal of Physics: Conference Series, 842,* 012079.

10. Altabey, W. A. (2017a). Delamination evaluation on basalt FRP composite pipe by electrical potential change. *Advances in Aircraft and Spacecraft Science, 4(5),* 515-528.

11. Altabey, W. A. (2017b). EPC method for delamination assessment of basalt FRP pipe: electrodes number effect. *Structural Monitoring and Maintenance, 4(1),* 69-84.

12. Altabey, W. A., Noori, M. (2018). Monitoring the water absorption in GFRE pipes via an electrical capacitance sensors. *Advances in Aircraft and Spacecraft Science, 5(4),* 411-434.

13. Persson, N., Fredrik, G., Markus, D. (2002). Indirect Tire Pressure Monitoring Using Sensor Fusion. *SAE 2002 World Congress & Exhibition, SAE Technical Paper 2002-01-1250.*

14. Luo, Q. (2014). *The research of indirect tire pressure monitor system based on the frequency method*. Wuhan University of Technology, Wuhan.

15. Han, Z., Song, J., Su, D. (2008). The design of a tire pressure monitoring and alarming system for vehicles based on tire rolling radius method. *Automotive Engineering*.

16. Han, Z., Liu, Q., Wang, L. (2010). Study on tire pressure monitoring and alarming system of automobile based on the comparison among standard pulse numbers. *China Mechanical Engineering*.

17. Liang, L., Gang, J., Xu, R., Jian, S., Kaihui, W. (2014). A variable structure extended kalman filter for vehicle sideslip angle estimation on low friction road. *Vehicle System Dynamics, 52(2),* 280-308.

18. Changzheng, W., Wei, Z., Quan, W., Xiaoyuan, X., Xinxin, L. (2012). TPMS (tire-pressure monitoring system) sensors: monolithic integration of surface-micromachined piezoresistive pressure sensor and self-testable accelerometer. *Microelectronic Engineering, 91,* 167-173.

19. Daniel, G., Oluremi, O., Salvatore, S., Mario, T. (2019). A real-time physical model for strain-based intelligent tires. *Sensors and Actuators A: Physical, 288,* 1-9.

20. Yu-Jen, W., Chung-De, C., Chung-Chih, L., Jui-Hsin, Y. (2015). A nonlinear suspended energy harvester for a tire pressure monitoring system. *Micromachines, 6,* 312-327.

21. Oche, A. E., Salem, C., David, B. (2019). An innovative decision rule approach to tyre pressure monitoring. *Expert Systems with Applications, 124,* 252-270.

22. Raul, G. L., Robert, B., Andrew, S. (2019). An in-wheel sensor for monitoring tire-terrain interaction: development and laboratory testing. *Terramechanics, 82,* 43-52.

23. Zhao, Y., Noori, M., Altabey, W. A., Naiwei, L. (2017). Reliability evaluation of a laminate composite plate under distributed pressure using a hybrid response surface method. *International Journal of Reliability, Quality and Safety Engineering, 24(3),* 1750013.

24. Altabey, W. A. (2018). High performance estimations of natural frequency of basalt FRP laminated plates with intermediate elastic support using response surfaces method. *Vibroengineering. 20(2),* 1099-1107.

25. Menglong, C., Dongyuan, Y. (2018). An improved D-S evidence theory in tire pressure monitoring system. *IOP Conference. Series: Materials Science and Engineering, 394,* 032112.

26. Binwen, H. (2013). Design of direct-type tire-pressure monitoring system based on SP37 sensor. *Sensors & Transducers, 160(12),* 74-79.

27. Cullen, J. D., Arvanitis, N., Lucas, J., Al-Shamma'a, A. I. (2002). In-field trials of a tyre pressure monitoring system based on segmented capacitance rings. *Measurement, 32(3),* 181-192.

28. Jingui, Q., Dong-Su, K., Dong-Weon, L. (2019). On-vehicle triboelectric nanogenerator enabled self-powered sensor for tire pressure monitoring. *Nano Energy, 49,* 126-136.

29. Hongjip, K., Wei Che, T., Jason, P., Lei, Z. (2019). Self-tuning stochastic resonance energy harvesting for rotating systems under modulated noise and its application to smart tires. *Mechanical Systems and Signal Processing, 122,* 769-785.

30. Jazar, R. (2014). *Vehicle dynamics: theory and application*, 2nd ed. Springer-Verlag New York.

31. Tan, D., Wang, Q. (2016). Modeling and simulation of the vibration characteristics of the in-wheel motor driving vehicle based on bond graph. *Shock and Vibration.* https://doi.org/10.1155/2016/1982390.

32. Liu, Y. (2008). Constructing equations of motion for a vehicle rigid body model. *SAE International Journal of Passenger Cars-Mechanical Systems, 1*. https://doi.org/10.4271/2008-01-2751.

33. NHTSA. (2016). IV. Tire pressure monitoring systems URL.
https://one.nhtsa.gov/cars/rules/rulings/tirepresfinal/tireprmonsys.html.

34. Transport & Environment. (2016). Failure of indirect tyre pressure monitoring systems puts drivers and road users at risk.
https://www.transportenvironment.org/sites/te/files/publications/2016_11_TPMS_report_final.pdf.

35. Wang, L., Zhang, Z., Yao, Y., Han, Z., Bin, W. (2017). Monitoring method of indirect TPMS under steering situation. *DEStech Transactions on Engineering and Technology Research*,
https://doi.org/10.12783/dtetr/mime2016/10229.

36. Altabey, W. A., Noori, M. (2017b). Fatigue life prediction for carbon fibre/epoxy laminate composites under spectrum loading using two different neural network architectures. *Sustainable Materials and Structural Systems, 3(1),* 53-78.

37. Altabey, W. A. (2017c). Prediction of natural frequency of basalt fiber reinforced polymer (FRP) laminated variable thickness plates with intermediate elastic support using artificial neural networks (ANNs) method. *Vibroengineering, 19(5),* 3668-3678.

38. Altabey, W. A. (2016a). FE and ANN model of ECS to simulate the pipelines suffer from internal corrosion. *Structural Monitoring and Maintenance, 3(3),* 297-314.

39. Altabey, W. A. (2016b). Detecting and predicting the crude oil type inside composite pipes using ECS and ANN. *Structural Monitoring and Maintenance, 3(4),* 377-393.

40. Altabey, W. A. (2016c). The thermal effect on electrical capacitance sensor for two-phase flow monitoring. *Structural Monitoring and Maintenance, 3(4),* 335-347.

41. Zhao, Y., Noori, M., Altabey, W. A., Awad, T. (2019). A comparison of three different methods for the identification of hysterically degrading structures using bwbn model. *Front. Built Environ, 4(80),* http://dx.doi.org/10.3389/fbuil.2018.00080.

42. Zhao, Y., Noori, M., Seyed, B. B., and Altabey, W. A. (2017). Mode shape based damage identification for a reinforced concrete beam using wavelet coefficient differences and multi-resolution analysis. *Structural Control and Health Monitoring, 25(3),* 1-41.

43. Zhao, Y., Noori, M., Altabey, W. A. (2017). Damage detection for a beam under transient excitation via three different algorithms. *Structural Engineering and Mechanics, 63(6),* 803-817.

44. Noori, M., Haifegn, W., Altabey, W. A., Ahmad, I. H. S. (2018). A modified wavelet energy rate based damage identification method for steel bridges. *International Journal of Science & Technology. Scientia Iranica, 25(6),* 3210-3230.

45. Ramachandran, P., Zoph, B., Quoc, V. L. (2017). Searching for activation functions. *Cornell University Library, Preprint*, 1710. 05941. http://arxiv.org/abs/1710.05941.

46. Yi, Z., Qi, L., Enhong, C., Yong, G., Zhao, J. L. (2014). Time series classification using multi-channels deep convolutional neural networks web-age information management. *15th International Conference on Web-Age Information Management.* WAIM 2014. *Lecture Notes in Computer Science, 8485*, Springer, Cham.
https://doi.org/10.1007/978-3-319-08010-9_33.

47. Diederik, P. K., Jimmy, B. (2014). Adam: a method for stochastic optimization. *Cornell University Library, Preprint,* 1412.6980, http://arxiv.org/abs/1412.6980.

48. Martinez-Cantin, R., de Freitas, N., Doucet, A., Castellanos, J. A. (2007). Active policy learning for robot planning and exploration under uncertainty. *Robotics: Science and Systems,* 321-328.

49. Brochu, E., Brochu, T., de Freitas, N. (2010). A bayesian interactive optimization approach to procedural

animation design. *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association,* 103-112.

50. Snoek, J., Larochelle, H., Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems,* 2951-2959.
51. Altabey, W. A. (2017d). An exact solution for mechanical behavior of BFRP nano-thin films embedded in NEMS, *Advances in Nano Research, 5(4),* 337-357.
52. Altabey, W. A. (2017e). A study on thermo-mechanical behavior of MCD through bulge test analysis. *Advances in Computational Design, 2(2),* 107-119.
53. Altabey, W. A. (2017f). Free vibration of basalt fiber reinforced polymer (FRP) laminated variable thickness plates with intermediate elastic support using finite strip transition matrix (FSTM) method. *Vibroengineering, 19(4),* 2873-2885.
54. Martinez-Cantin, R. (2014). Bayesopt: a bayesian optimization library for nonlinear optimization, experimental design and bandits. *Machine Learning Research, 15 (1),* 3735-3739.
55. Hutter, F., Hoos, H. H., Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. *Learning and Intelligent Optimization,* 507-523.
56. Bergstra, J., Bardenet, R., Bengio, Y., K´egl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems,* 2546-2554.
57. Bergstra, J., Yamins, D., Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Proceedings of the 30th International Conference on Machine Learning*, 115-123.
58. Ian, D., Michael, M., Scott, C., Patrick, H., Alexandra, J. et al. (2016). A stratified analysis of bayesian optimization methods. *Cornell University Library, Preprint,* 1603.09441. http://arxiv.org/abs/1603.09441.
59. Georg, D. (1996). Neural networks for time series processing, neural network world. http://dx.doi.org/10.1.1.45.5697.
60. Karn, U. (2016). An intuitive explanation of convolutional neural networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.
61. Zhao, Y., Noori, M., Altabey, W. A., Ghiasi, R., Wu, Z. (2018b). Deep learning-based damage, load and support identification for a composite pipeline by extracting modal macro strains from dynamic excitations. *Applied Sciences, 8(12),* 2564.
62. Wang, T., Noori, M., Altabey, W. A., Zhao, Y. (2019). Identification of cracks in an euler-bernoulli beam using bayesian inference and closed-form solution of vibration modes, under review. *Smart Materials and Structures.*
63. Ghannadi, P., Kourehli, S. S., Noori, M., Altabey, W. A. (2019). Structural damage detection and severity identification using mode shape expansion and grey wolf optimizer. Under Review, *Inverse Problems.*
64. Olah, C. (2015). Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
65. Zachary, C. L., John, B., Charles, E. (2015). A critical review of recurrent neural networks for sequence learning. *Cornell University Library, Preprint*, 1506.00019. http://arxiv.org/abs/1506.00019.
66. John, C., Borges, G. (2017). Deep learning for time-series analysis. *Cornell University Library, Preprint,* 1701.01887. http://arxiv.org/abs/1701.01887.
67. Adit, D. (2015). A beginners guide to understanding convolutional neural networks. *A Beginners Guide to Understanding Convolutional Neural Networks-Adit Deshpande-CS Undergrad at UCLA (19).* https://adeshpande3.github.io/A-Beginner's-Guide-To-UnderstandingConvolutional-Neural-Networks/.
68. Wiggins, V., Engquist, S., Looper, L. (1992). Neural network applications: a literature review. Tech. rep. Air Force. http://www.dtic.mil/dtic/tr/fulltext/u2/a258148.pdf.
69. Alippi, C., de Russis, C., Piuri, V. (2003). A neural-network based control solution to air-fuel ratio control for automotive fuel-injection systems. *IEEE Transactions on Systems,* 259-268.
70. Jiang, T., Petrovic, S., Ayyer, U., Tolani, A., Husain, S. (2015). Self-driving cars: disruptive or incremental. *Applied Innovation Review, 1*. http://cet.berkeley.edu/wp-content/uploads/Self-Driving-Cars.pdf.
71. Habib, K. (2017). Automatic vehicle control systems. PE 16-007 Tech. rep. NHTSA URL. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.pdf.

72. Hollemans, M. (2016). Convolutional neural networks on the iphone with vggnet. http://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/.

73. Ng, A. (2019). Model selection and train/validation/test sets, coursera: machine learning. lecture notes. https://www.coursera.org/learn/machine-learning/supplement/XHQqO/model-selection-and-train-validation-test-sets.

74. Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures, *Cornell University Library, Preprint,* 1206.5533.

75. Phaisangittisagul, E. (2016). An analysis of the regularization between L2 and dropout in single hidden layer neural network. *7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS), Bangkok, Thailand, 174*. http://dx.doi.org/10.1109/ISMS.2016.14.

76. Ioffe, S., Christian, S. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. *Cornell University Library, Preprint,* 1502.03167. http://arxiv.org/abs/1502.03167.

77. James, D. M. (2013). Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training. https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for.

78. Karim, F., Majumdar, S., Darabi, H., Chen, S. (2018). LSTM fully convolutional networks for time series classification. digital object identifier. *IEEE Access, 6,* 1662-1669.

79. Andrew, G. (2014). Math [Code], Bringing HPC Techniques to Deep Learning. http://andrew.gibiansky.com/blog/machinelearning/convolutional-neural-networks/.

80. Fawaz, H., Germain, F., Jonathan, W., Lhassan, I., Pierre-Alain, M. (2018). Deep learning for time series classification: a review. *Cornell University Library, Preprint, 1809.04356.* https://arxiv.org/abs/1809.04356v4.

81. Daniel III, W. V., Daniel W. V. (2012). Individual vehicle data search service Tech. rep. 4N6XPRT Systems La Mesa CA 91942. http://www.4n6xprt.com/Crash-Test-4/ARC-CSI_2012_Conference-Rio_&_Yaris_data_packet_data.pdf.

82. Overton, J., Mills, B., Ashley, C. (1969). The vertical response characteristics of the non-rolling tyre. *Proceedings of the Institution of Mechanical Engineers, Automobile Division 1947-1970, 184 (1),* 25-40.

83. Dixon, J. (2007). *The Shock Absorber Handbook.* Wiley-PEPublishing Series (John Wiley & Sons).

84. Giaraffa, M. (2017). Tech Tip: Springs & Dampers, Part Three, Revenge of the Damping Ratio. http://www.optimumg.com/technical/technical-papers/.

Appendix A. Derivations of Simulation Input Parameters

All constants used as simulation input variables are derived as follows. Except for identifying, all of these calculations are performed in MATLAB.

m_s is simply taken from Daniel III and Daniel [81] and divided by 4 to account for the quarter-car model.

$$m_s = 1109kg/4 = 277.25kg \tag{A1}$$

Assuming that the tire in use across all vehicles is a radial-ply 165×13 tire (a very common tire size found on most passenger vehicles), a linear model for static stiffness based on tire inflation pressure can be used [82].

The model is graphically presented in Fig. A1 and expressed by Eq. (A2). The model is only accurate above 15 psi-an acceptable limitation as 15 psi is well below the threshold for "under-pressurized".

$$k_s = 30185pu + 46.375 \tag{A2}$$

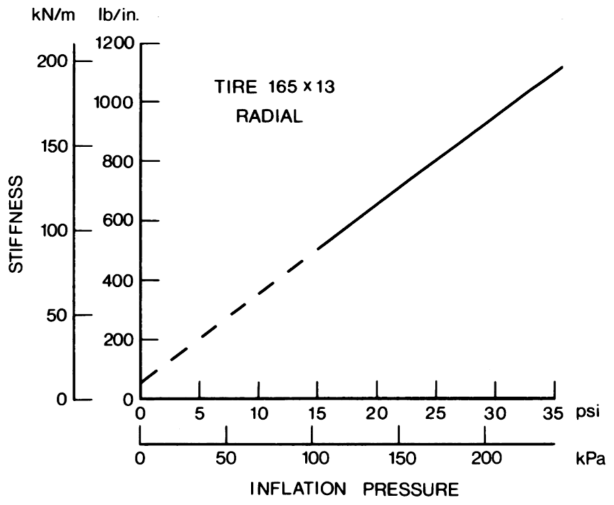


Figure A1: Static stiffness *vs.* inflation pressure for a radial-ply car tire [82]

Utilizing the average quarter-car ratio of the sprung to unsprung masses and the one can identify the expected value for mu [Jazar (2014)]:

$$\varepsilon = \frac{m_s}{m_u} = 8 \Rightarrow m_u = \frac{m_s}{\varepsilon} = \frac{277.25kg}{8} = 34.69kg \tag{A3}$$

It should be noted that mu should vary with tire pressure due to the additional air inside the tires. However, the mass of the air is insignificant relative to the rest of the unsprung mass (< 0.1%). Nonetheless, the mass of the air is calculated and included in the unsprung mass for these simulations. The calculations are performed in MATLAB.

To identify the suspension's stiffness and damping coefficients, assume that the suspension is tuned for a properly-inflated tire. With $pu = 32psi$, Eqs. (A2) and (A3) give the $k_u = 6979.53kPa$ and $m_u = 34.69kg$ respectively. With these values, one can find the natural frequency of the unsprung mass ω_u:

$$\omega_u = \sqrt{\frac{k_u}{m_u}} = \sqrt{\frac{6979.53\frac{N}{m}}{34.69kg}} = 448.612\frac{rad}{\sec} \tag{A4}$$

The average quarter-car ratio for sprung and unsprung natural frequencies is used to identify the sprung mass's natural frequency ω_s.

$$\alpha = \frac{\omega_s}{\omega_u} = 0.1 \Rightarrow \omega_s = \alpha\omega_u = (0.1)\left(448.612\frac{rad}{\sec}\right) = 44.86\frac{rad}{\sec} \tag{A5}$$

We already know that $m_s = 277.25kg$, so identifying k_s is trivial.

$$\omega_s = \sqrt{\frac{k_s}{m_s}} \Rightarrow k_s = \omega_s^2 m_s = \left(44.86\frac{rad}{\sec}\right)^2 (277.25kg) = 557.97\frac{N}{m} \tag{A6}$$

To calculate c_s, we can use the relationship between ω_s and the damping ratio $\zeta = \frac{c_s}{c}$ where c is the critical damping coefficient. Numerous sources suggest the proper damping ratio in passenger vehicles to be between 0.2 and 0.3 [83,84]. For this work, we define ζ = 0.25.

$$\zeta = \frac{c_s}{2m_s\omega_s} \Rightarrow c_s = 2\zeta m_s\omega_s = 2(0.25)(277.25kg)\left(44.86\frac{rad}{\sec}\right) = 6218.8\frac{N-\sec}{m} \tag{A7}$$