Machine Learning on Embedded Systems: An Exploratory Study on Gas Chromatography Analysis

Presented at Pittcon 2020

Tismo Technology Solutions (P) Ltd
Bangalore, India

ABSTRACT

This paper proposes a novel approach of utilizing Machine Learning (ML) models for embedded systems. This is demonstrated through the deployment of an ML model on a Gas Chromatography (GC) system, used for natural gas analysis. Unlike other schools of research that treat the analysis of chromatograms as image classification problems, this study treats chromatograms as time-series inputs. The trained model is ported to C programming language and deployed on an embedded system. Experimental results demonstrate the superior accuracy and robustness achieved by this ML based approach, when compared to the traditional analysis using heuristic algorithms. The execution time of the ML model on the embedded system is also found to be on par with analysis using heuristic algorithms.

INTRODUCTION

The popular perception of Machine Learning (ML) is as a set of complex algorithms that can only be applied on systems with high processing capabilities. On the contrary, ML has remarkable potential to enhance the capabilities of instruments. However, its application on embedded systems is yet to be fully explored.

Prevalent literature indicates that ML models can increase the accuracy and robustness of analyses, over a variety of applications. In existing measurement systems such as Mass Spectrometry [1] [2] and Chromatography [3] [4] [5], ML models can provide increased accuracy and robustness against process variations. In control systems, where prevailing algorithms are not sufficiently accurate to depict complex real-world systems, ML models can be used to increase robustness and to remove simplifying assumptions[6] [7]. The automated monitoring, detection and identification of ECG signals can be done by a real-time, accurate and robust analysis using the ML approach [8] [9] [10] [11]. ML has shown great promise in detecting potential motor anomalies due to bearing faults, through both current analysis and vibrational analysis techniques [12] [13] [14] [15] [16] [17]. While almost impossible to detect visually by manual inspection or with spectral analysis, researchers have been able to detect bearing faults with close to 100% accuracy using ML models. Researchers using an ML based approach have recently achieved high performance levels in terms of detection and location of structural damage directly from the raw vibration signals, without the need for data pre-processing [18] [19] [20] [21] [22].

In this study, ML is proposed as a viable option for use in embedded systems. This is illustrated using an ML model to analyze the constituents of a natural gas sample in a Gas Chromatography (GC) system. In a GC system, variations in several key parameters such as retention

Copyright © 2020 Tismo Technology Solutions (P) Ltd

time, temperature etc., complicates the analysis model. Using ML on GC has enhanced the accuracy of analysis and has shown promise in enhancing the reliability of the analysis process, by making the study immune to variations in flow rate, retention time, inlet pressure and outlet pressure, viscosity and velocity of the carrier gas, detector temperature, oven temperature, sample size etc. The execution time and memory consumption of the ML model was found to be viable for deployment on an embedded system.

Most of the research in this field has been directed at treating chromatograms as image classification problems, using 2D CNNs [3] [23] [24]. However, this study treats chromatograms as time-series inputs. This is a novel approach to peak detection and evaluation, that optimizes the accuracy of the analysis.

This paper explores the premise that ML models can impart a level of intelligence to embedded systems, that would otherwise be difficult to implement, using traditional approaches. This is demonstrated through the deployment of an ML model for analysis on a GC system.

ML FOR EMBEDDED SYSTEMS

An ML model for an embedded system is developed in two distinct phases - design and deployment. The design phase includes identifying parts of the system that undertake complex data analysis, collating the training data sets, developing the suitable ML model, training, validating and testing the model.

The deployment phase involves porting the model to a low-level language like C/C++ and optimizing the ML model for the embedded system by eliminating redundant computations. Calibration is then applied to the output of the model.

Data for training is acquired during product development and prototyping and can be augmented to create training data sets. Large amounts of data are used to train the model in a high-performance computational environment. Training is the process of automatically determining the optimum variables and functions for error minimization. This is done by evaluating the loss function of the model. The functions and variables that form the model are iteratively updated through an optimization algorithm. Validation is the process of frequently evaluating the trained model, to fine tune its hyperparameters. This is done in tandem with training. Once the loss function converges to an acceptable level of error, the model is said to be trained. Testing is the process of evaluating the performance of the trained model against a previously unseen set of data.

Once an ML model has been created, it can be deployed to an embedded system. The model is ported to a low-level language like C/ C++. The functions and variables that form the model are optimized by reducing memory allocations and redundant computations. This would serve as the analysis logic of the system. After the ML model is deployed, factory and user calibration must be performed to account for the instrument specific variations.

Copyright © 2020 Tismo Technology Solutions (P) Ltd

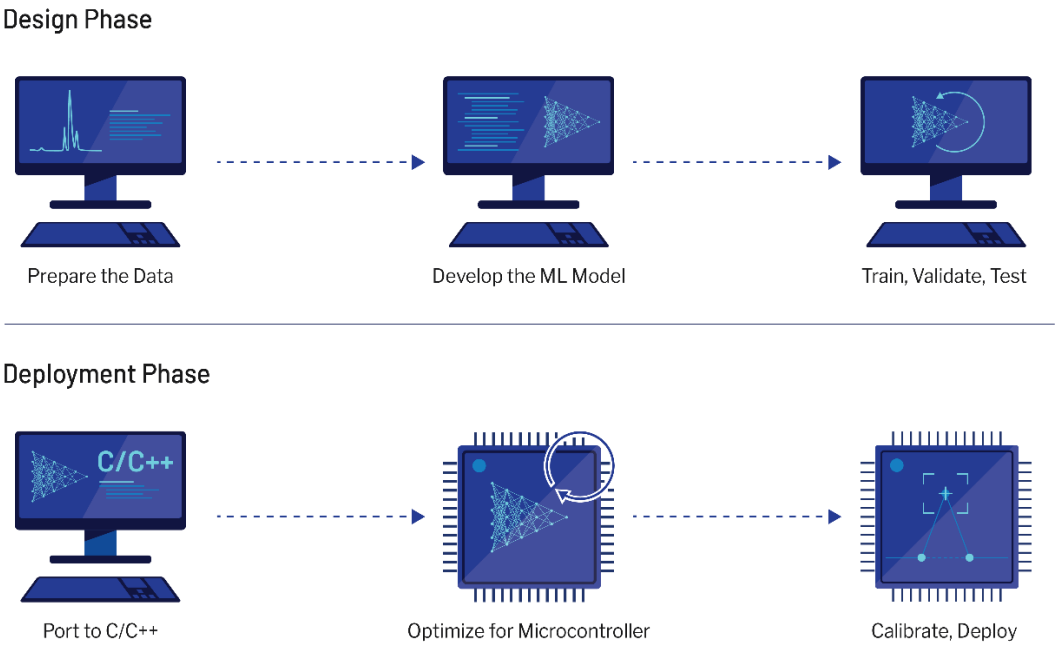

Figure 1: Design and Deployment of ML on Embedded Systems

METHODOLOGY

For the purpose of this study, an ML model was deployed on a GC system used to analyze the thermal characteristics of Natural Gas. A Multivariate Regression model was employed. This is a regression model that generates multiple outputs and is trained by a supervised learning algorithm. Chromatograms with known results were used to create, train and validate the ML model on a PC. This model was then ported to C and its performance was evaluated on the embedded system.

GC is one of the most popular separation techniques, used for the analysis of samples that vaporize without decomposition. The gaseous samples under analysis adsorb on a column at different rates. This causes each constituent to elute at different times. This is known as the retention time of the constituent. The GC system used in this study features a 32-bit microcontroller responsible for process management, oven temperature control and data acquisition. It also provides supervisory control, carries out data analysis and is responsible for the HMI. Thermal Conductivity Detector (TCD) was used to provide extremely high sensitivity. A 24-bit ADC is used for the TCD measurements.

Technology

For the purpose of this study, Python 3.0, Keras, an open-source neural-network API written in Python and TensorFlow 2.0 libraries were utilized. Here, it is important to highlight that the ML model was trained on a high-performance system. Once the model was finalized, it was optimized and ported to C programming language, to be deployed on the GC system.

Machine Learning Model

The ML model in this study uses One Dimensional Convolutional Neural Network (1D CNN) layers. This is a type of deep neural network that performs a single dimensional array operation called convolution, while mapping its input to the output.

A GC system acquires data points indexed in time, called chromatograms. The chromatogram is analyzed to

Copyright © 2020 Tismo Technology Solutions (P) Ltd

identify and quantify constituents in samples. For such time-series applications, 1D CNNs show a lot of promise. 1D CNNs require only linear array operations. This implies lower memory requirements and lower computational complexity, as compared to 2D CNNs. This makes them suitable for real-time, low-cost hardware implementations like embedded systems.

The ML model used in this study is composed of one Dynamic Compression layer, one Average Pooling layer, four Convolutional layers and three Dense layers, in that order. Each node in a layer is connected to some or all the nodes of the previous layers.

The Dynamic Compression layer was introduced to scale the magnitude of the data points to a range between -1 and +1. Both the chromatogram data and the corresponding results were compressed. This led to faster convergence of error, during training. An Average Pooling layer is a moving average filter, introduced to reduce white noise in the data obtained from the detector. The Convolutional layers performed a specialized type of linear operation. An array of weights, called the kernel, is applied on all elements of the input array, in batches called strides. A product sum between the kernel and a group of input elements is computed for each output position. Each filter in the Convolutional layer can be thought of as identifying a particular pattern or feature. The outputs are then passed through a nonlinear activation function, which in this study, is the Rectified Linear Unit (ReLU). The Dense Layers perform weighted summation operations on the outputs of the previous layers. The outputs of the dense layers (except the outputs of the final layer) are then passed through the ReLU activation function. The results need to be decompressed and calibration applied. The percentage values of the constituent gases are then calculated.

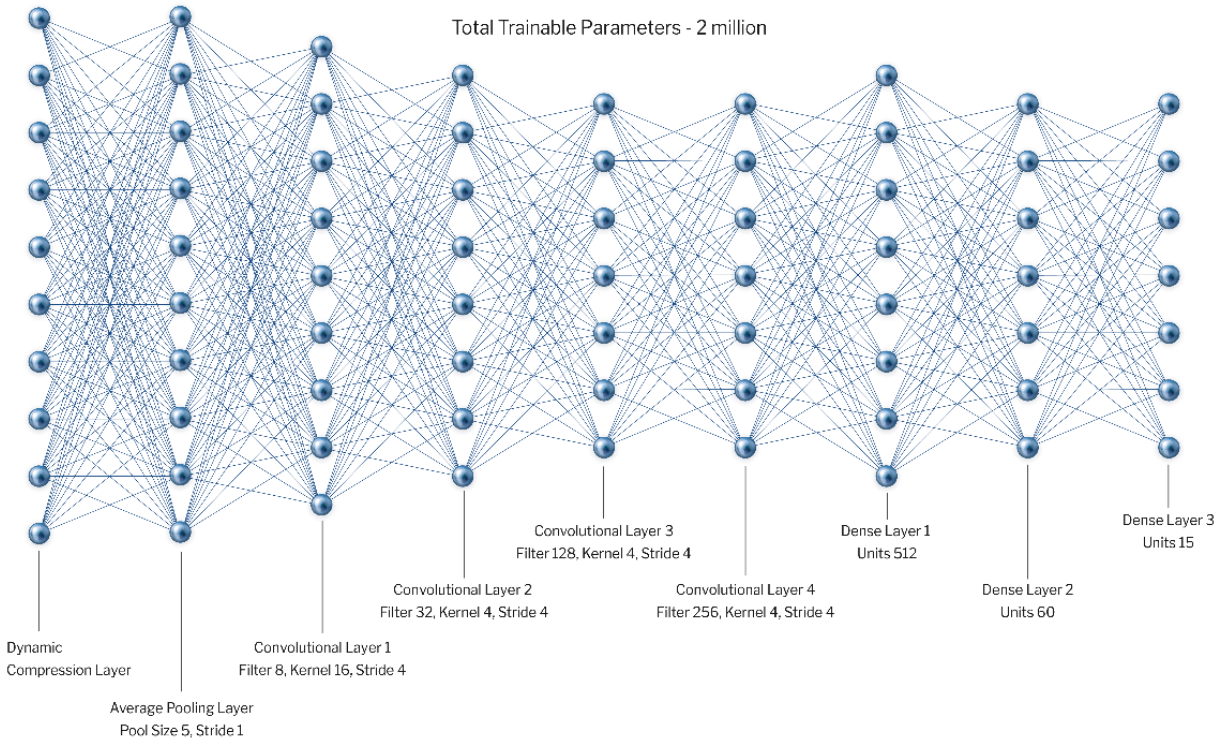

Figure 2: Machine Learning Model used in this Study

Copyright © 2020 Tismo Technology Solutions (P) Ltd

Data Preparation

A set of 50 chromatograms was initially available for this study. Data augmentation was carried out through various techniques to simulate process variations such as linear time shifting (± 1 min), time scaling (± 1 min), removal of the constituent gases and combinations of each of these options. This generated a total of about 70,000 chromatograms, with their corresponding results. These were divided randomly into the training set (about 56,000 chromatograms), validation set (about 7,000 chromatograms) and the test set (about 7,000 chromatograms).

A training set is a large set of chromatograms, used to train the ML model. A validation set is a set of chromatograms, used to fine-tune the hyperparameters (such as number of layers, configuration of each layer with parameters like kernel size, stride etc.). A test set is an independent set of chromatograms that is used only to evaluate the performance of the final model.

Training the Model

Training is the process of minimizing the loss function by automatically determining optimum values for all learnable parameters (weights and biases). A loss function measures the deviation between output predictions of the network and known results. In this study, the Mean Squared Error (MSE) loss function was used. MSE computes the average squared difference between the estimated values and the actual values. Loss values are calculated during forward propagation and learnable parameters are updated during backpropagation.

Adaptive Moment Estimation (Adam) [25] methodology was the optimization algorithm used during backpropagation. Adam is a variation of Stochastic Gradient Descent (SGD). Adam estimates the first and second moments of the gradient to adjust the learning rate for each weight of the neural network.

Validation and Testing

Training of the ML model was stopped when the MSE on the validation set converged. Convergence was observed at epoch count (number of training iterations) of 50. The trained model was then tested with a test set of 7,000 chromatograms and an average MSE of 0.09 was observed.

The superior performance of the ML algorithm is evident across each of the sample constituents. The table below captures the detected percentage of each constituent, for an arbitrary chromatogram using the ML model. Mean Squared Error and Root Mean Squared Error were calculated.

Constituents	Actual (in Mole %)	ML (in Mole %)
C6+	0.080	0.10
Nitrogen	1.479	1.57
Methane	91.139	91.16
CO_2	1.349	1.36
Ethane	3.966	4.01
Propane	0.979	1.07
i-Butane	0.300	0.29
n-Butane	0.301	0.30
i-Pentane	0.150	0.17
n-Pentane	0.150	0.16
neo-Pentane	0.100	0.10
Mean Square Error		**0.0018**
Root Mean Square Error		**0.0427**

Table 1: Analysis results of a Single Chromatogram using ML Algorithm

Copyright © 2020 Tismo Technology Solutions (P) Ltd

Deployment on Embedded System

The trained ML model was ported to C programming language. A Python script was used to extract the final values of weights and biases of each layer and to generate C files. The operations of the ML model were then replicated through a C program that utilized the final values of weights and biases. This resulted in an ML model that eliminated unnecessary memory allocations and computations and was optimized to run on an embedded system. No external libraries or frameworks were used to run the model. Calibration was applied to the output of the model. The system was tested using multiple samples and results were validated against the known values of reference samples.

This study found that the ML model consumed about 8 MB of memory. The table below illustrates the performance parameters of the ML model on a PC and various embedded systems, examined over the course of this study.

Platform	RAM	CPU	Average Time (s)
Windows 10 PC	8 GB	Intel i3 @ 3.9 GHz	0.12
F&S ArmStone A9	1 GB	NXP i.MX 6 ARM Cortex-A9 (max 1.2 GHz)	1.17
Beaglebone Green	512 MB	AM335x 1GHz ARM Cortex A8	2.29
Raspberry Pi 3B+	1 GB	BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz.	1.24

Table 2: Performance Parameters of the ML model

RESULTS

An ML model that accurately analyzed the constituents of a natural gas sample through GC, was realized. The model was deployed on the embedded system. Accuracy of the GC analysis was dramatically improved using the ML model, when compared to the traditional analysis. The Absolute Errors recorded by the ML algorithm and the traditional algorithm for 64 arbitrary chromatograms, are captured in the scatter plot below. Error was calculated by comparing the values of concentration of constituents detected by the algorithms to the known calibrated values.

Copyright © 2020 Tismo Technology Solutions (P) Ltd

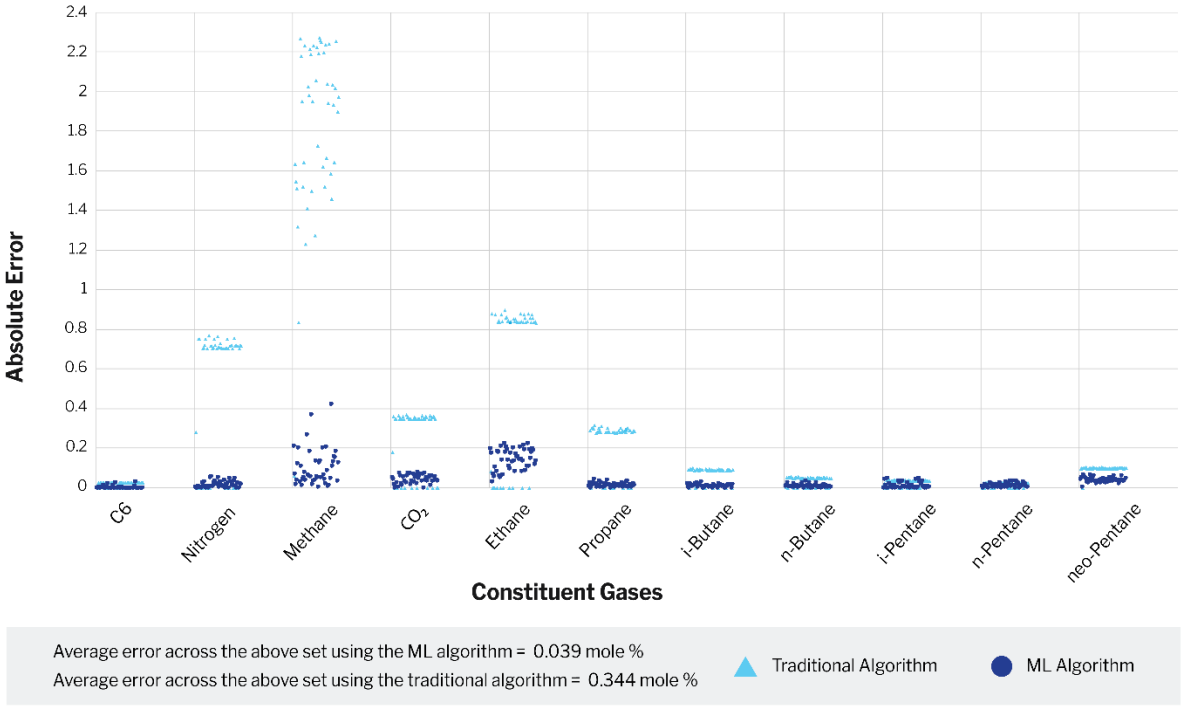

Figure 3: Scatterplot comparing the accuracy of GC analysis using ML Algorithm and Traditional Algorithm

DISCUSSION

This exploratory study indicates that ML is effective in embedded system applications, where a multitude of variables and experimental conditions make it difficult to arrive at an all-encompassing analysis model. ML is found to be a viable option to increase accuracy and robustness of data analysis in embedded systems.

The traditional approach to GC analysis involves noise filtering, baseline correction, peak detection, peak selection, peak identification, peak area computation and quantification. Each of these steps introduces errors that cascade and lead to a system that is susceptible to process variations. On the contrary, the novel ML approach used in this study identified and quantified gas constituents with increased accuracy and was robust against process variations. Moreover, porting the ML model to C resulted in the creation of a model optimized to run on an embedded system.

For training, this approach would need a high-performance system and design time effort comparable to any typical software development for similar applications. Once a model is finalized, it can be ported to embedded systems. Alternatively, this can be automated through platforms like TensorFlow Lite. However, manual porting is the recommended approach, as it would eliminate dispensable memory initializations and redundant computations.

This study establishes the viability of deploying ML models to embedded systems, paving the way for intelligent analysis. This study also indicates that ML models developed using 1D CNNs are ideal for analyzing time-series data. The findings of this study indicate that using ML to treat chromatograms as time-series inputs instead of images, could be the next step in enhancing the accuracy and robustness of GC analysis.

Copyright © 2020 Tismo Technology Solutions (P) Ltd

Future work in this area can be to further optimize the model by reducing its complexity without compromising on accuracy. Furthermore, it may be possible to redesign the GC system for a faster flow rate, faster GC start-up and a compressed chromatogram, assuming that the ML model will be able to achieve similar accuracy. In the future, this work can also be extended to Liquid Chromatography (LC) and Gas Chromatography-Mass Spectrometry (GC-MS) applications.

REFERENCES

1. *Georgios A Theodoridis, Helen G Gika, Elizabeth J Want, and Ian D Wilson. Liquid chromatography-mass spectrometry based global metabolite profiling (2012).*

2. *C A Smith, E J Want, G O'Maille, R Abagyan, and G Siuzdak. XCMS: processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification (2006).*

3. *Ralf Tautenhahn, Christoph B¨ottcher, and Steffen Neumann. Highly sensitive feature detection for high resolution LC/MS (2008).*

4. *A. Smolinska, A. C. Hauschild, R. R. Fijten, J. W. Dallinga, J. Baum-bach, and F. J. van Schooten, Current breathomics-a review on data pre-processing techniques and machine learning in metabolomics breath analysis (2014).*

5. *Hong Yang, Zbigniew Ring, Yevgenia Briker, Norma McLean, Wally Friesen, Craig W. Fairbridge, Neural network prediction of cetane number and density of diesel fuel from its chemical composition determined by LC and GC–MS (2002).*

6. *Antonelli, G.: Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems (2009).*

7. *Duriez, T., Brunton, S.L., Noack, B.R.: Machine learning control (MLC). – Taming Nonlinear Dynamics and Turbulence (2017).*

8. *S. Kiranyaz, T. Ince, M. Gabbouj, Personalized Monitoring and Advance Warning System for Cardiac Arrhythmias (2017).*

9. *S. Kiranyaz, T. Ince, R. Hamila, M. Gabbouj, Convolutional Neural Networks for patient-specific ECG classification (2015).*

10. *S. Kiranyaz, T. Ince, M. Gabbouj, Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks (2016).*

11. *S. Kiranyaz, T. Ince, M. Gabbouj, Personalized Monitoring and Advance Warning System for Cardiac Arrhythmias (2017).*

12. *T. Ince, S. Kiranyaz, L. Eren, M. Askar, M. Gabbouj, Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks (2016).*

13. *S. Kiranyaz, A. Gastli, L. Ben-Brahim, N. Alemadi, M. Gabbouj, Real-Time Fault Detection and Identification for MMC using 1D Convolutional Neural Networks (2018).*

Copyright © 2020 Tismo Technology Solutions (P) Ltd

14. *O. Abdeljaber, S. Sassi, O. Avci, S. Kiranyaz, A.A. Ibrahim, M. Gabbouj, Fault Detection and Severity Identification of Ball Bearings by Online Condition Monitoring (2018).*

15. *L. Eren, T. Ince, S. Kiranyaz, A Generic Intelligent Bearing Fault Diagnosis System Using Compact Adaptive 1D CNN Classifier (2019).*

16. *L. Eren, Bearing fault detection by one-dimensional convolutional neural networks (2017).*

17. *W. Zhang, C. Li, G. Peng, Y. Chen, Z. Zhang, A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load (2018).*

18. *O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, D.J. Inman, Wireless and real-time structural damage detection: A novel decentralized method for wireless sensor networks (2018).*

19. *O. Avci, O. Abdeljaber, S. Kiranyaz, D. Inman, Structural Damage Detection in Real Time: Implementation of 1D Convolutional Neural Networks for SHM Applications (2017).*

20. *O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, D.J. Inman, Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks (2017).*

21. *O. Avci, O. Abdeljaber, S. Kiranyaz, B. Boashash, H. Sodano, D.J. Inman, Efficiency Validation of One Dimensional Convolutional Neural Networks for Structural Damage Detection Using a SHM Benchmark Data (2018).*

22. *O. Abdeljaber, O. Avci, M.S. Kiranyaz, B. Boashash, H. Sodano, D.J. Inman, 1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data, Neurocomputing (2017).*

23. *K. H. Liland, T. Almoy, and B. H. Mevik, Optimal choice of baseline correction for multivariate calibration of spectra (2010).*

24. *D. Cirean, U. Meier, and J. Schmidhuber, Multi-column Deep Neural Networks for Image Classification, (2012).*

25. *Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization (2015).*

Copyright © 2020 Tismo Technology Solutions (P) Ltd