Rank Reduction of LSTM Models for Online Vibration Signal Compensation on Edge Computing Devices

Josh McGuire[a,b], Joud N. Satme[a], Daniel Coble[a,c], Austin R.J. Downey[a,d], Jason Bakos[b], Ryan Yount[a], and Arion Pons[e]

[a]Department of Mechanical Engineering, University of South Carolina, SC, USA 29208
[b]Department of Computer Science and Computer Engineering, University of South Carolina, SC, USA 29208
[c]Department of Mechanical Engineering and Materials Science, Duke University, NC, USA 27708
[d]Department of Civil and Environmental Engineering, University of South Carolina, SC, USA 29208
[e]Department of Mechanics and Maritime Sciences, Chalmers University of Technology, Gothenberg, Sweden

ABSTRACT

There is increasing demand for the deployment of machine learning models on lightweight microcontrollers and internet of things devices in situations where off-device processing is impractical. Deploying models on such devices can be challenging due to performance and memory constraints, as the memory footprint and the number of operations necessary for inference scale rapidly with model size. This study proposes a technique that employs singular value decomposition to reduce the number of weights in long short-term memory models while having a minimal impact on their accuracy, leading to improvements in both memory footprint and performance. Furthermore, the rank reduction process results in dense matrices, which are more computationally efficient when compared to sparse matrices. The proposed technique is particularly advantageous for deployment on microprocessors like the Teensy 4.0's Arm Cortex-M7 CPU, which, despite having a floating-point unit, lacks significant parallelization features, making it an ideal candidate for demonstrating the benefits of rank reduction in real-world applications. We validate the method by training a long short-term memory model to perform non-linear signal compensation on micro-electromechanical systems accelerometer data, extending low-frequency resolution without requiring hardware upgrades. Training data was collected using a reference accelerometer and a function-generator-driven electromagnetic shaker. Performance evaluation shows that the rank-reduced long short-term memory model achieves a 16.5% reduction in parameter count (from 10,451 to 8,861 parameters) while maintaining nearly identical signal fidelity. The reduced model achieved a signal-to-noise ratio of 14.8498 dB compared to 14.0765 dB for the original. This is a key step towards the first known deployment of a deep-learning-based signal compensation technique on a drone-deployable structural health monitoring sensor package, showcasing the potential for scalable, efficient long short-term memory model deployment on edge devices in infrastructure monitoring applications.

Keywords: edge computing, error compensation, memory-constrained devices, LSTM model, rank reduction

Further author information: (Send correspondence to Austin Downey)
Austin Downey: Email: austindowney@sc.edu

1. INTRODUCTION

The growing prevalence of internet of things (IoT) devices such as the ones used in structural health monitoring (SHM) creates a significant strain on data processing resources. While many IoT devices transmit data to be processed offline, the overwhelming quantities of data being produced make data processing at the edge essential.[1] IoT devices, with their requirement to be low-cost and power-efficient, are heavily constrained computationally. Their microcontrollers and limited memory are sufficient for simple data processing algorithms, but struggle with the high computational demands imposed by machine learning inference.[2] As machine learning develops into an integral part of data processing, solutions to this challenge are highly sought after.

An approach that has proven to be essential to deploying ML models on existing edge devices is model compression, which enables the use of models that would otherwise have been too large to fit into memory. Various model compression strategies have been developed with the goal of lowering the model's memory footprint or improving inference latency; among the most significant are quantization, pruning, and low-rank approximation.[3] Despite their prevalence, improved techniques are still sought after.

We propose a generalizable improvement to low-rank approximation using a degree-of-freedom-based technique to efficiently compute the matrix-vector product of reduced-rank matrices. The proposed technique yields both memory and computation savings immediately while preserving bit precision and dense matrix structure. To demonstrate the efficacy of this technique, we apply it to the non-linear signal compensation problem using a Long-Short Term Memory (LSTM) network. We then deploy the model to a UAV-deployable SHM sensing node and compare the performance between the uncompressed model and its compressed counterpart using signal-to-noise ratio (SNR), memory footprint, and inference latency as a metric.

The contributions of this work are twofold. First, the proposed degree-of-freedom-based technique significantly improves the viability of low-rank approximation, providing a compression option that preserves bit precision and dense matrix structure. Second, we provide a crucial step towards the first known deployment of the deep-learning-based signal compensation technique proposed by Satme et al.[4] on an SHM sensing node.

2. BACKGROUND

This section describes the relevant background topics for this work.

2.1 Quantization and Pruning

Quantization and pruning have become standard methods of model compression, seeing widespread deployment across a variety of applications.[5] Quantization is the process of converting fixed point values to lower-precision integers, which can provide a significant reduction to the model's memory footprint. Pruning refers to the elimination of weights close to zero, reducing the memory footprint of the model by omitting them.[6]

Despite the prevalence of these methods, their associated tradeoffs encourage the development of other techniques. Quantization provides a lower memory footprint, but does little to reduce the computational complexity of inference on general-purpose computers, especially if those computers are equipped with floating-point units (FPU). Similarly, the supporting data structures necessary for sparse matrix computations can cause a significant increase in latency, as they are more likely to cause more cache misses, increasing the amount of time being spent waiting for memory accesses.[7] Optimal deployment of models compressed using these techniques would involve the use of hardware accelerators that are not present on the majority of IoT devices. These limitations encourage the development of other methods that would be more performant on general-purpose computers.

2.2 SVD Based Low-Rank Approximation

The low-rank approximation problem describes the challenge of representing the information in a matrix using another matrix of a lower rank. That is, for a $m \times n$ matrix A and desired rank $r \leq \min(m, n)$, to find the rank r matrix $\hat{A}$ which minimizes the difference $\|A - \hat{A}\|_X$, in some matrix norm X. For a large class of the most popular norms, the answer is the truncated singular value decomposition. Where $U\Sigma V^{\mathrm{T}} = A$ is the SVD, this is the matrix that zeros out the $\min(m, n) - r$ smallest singular values from the diagonal matrix Σ. Along this motivation, we restrict ourselves to eliminating the ranks of singular vectors, by setting the corresponding singular values to zero.

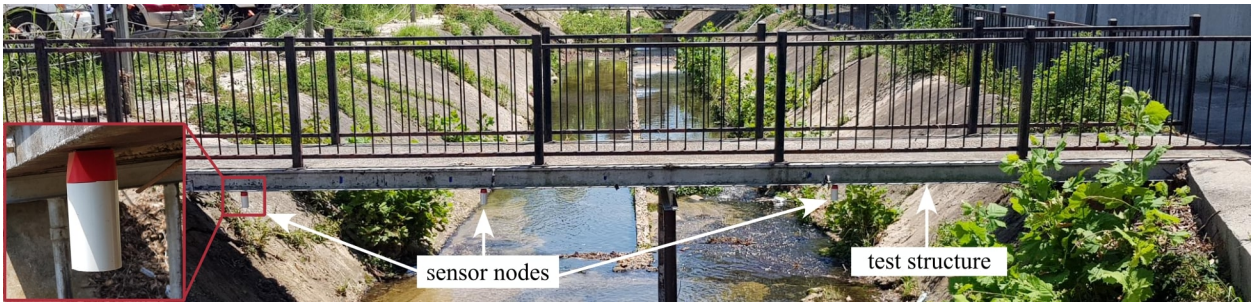

Figure 1. Field deployment of a sensor package network for structural health monitoring on a pedestrian bridge.

Direct deployment of a reduced singular value decomposition as done by Winata et al.[8] does not produce a reduction in computation until a threshold amount of rank reductions is performed to offset the cost of doing the two matrix-vector multiplies (assuming that Σ is rolled into either U or V^{T}), as such a scheme would require

$$mr + rn \text{ multiplies, and} \tag{1}$$

$$(mr - r) + (rn - n) \text{ adds.} \tag{2}$$

For a square matrix $m = n$, the break-even point is after $\lceil m/2 \rceil$ rank reductions are performed. This limits the applicability of low-rank approximation for edge machine learning, as the lightweight models deployed on edge lack the high redundancy found in large, fully-connected models.[9] It is therefore unlikely enough reductions can be performed before error rises to an unacceptable level.

2.3 Vibration Signal Compensation for SHM Sensing Nodes

The edge systems deployed for SHM are required to be cost-efficient and non-invasive, leading to compromises being made in the quality of data recorded. The low-frequency vibrations of structures such as the bridge shown in Figure 1 often fall below the usable range of the accelerometers used in these devices, which motivated the need for higher resolution sensing nodes.[10] However, a cost-effective solution using existing hardware is desirable.

Micro-electromechanical systems (MEMS) accelerometers have proven to be a good balance between cost and performance, which is desirable for SHM applications.[11] To further enhance the low-frequency detection capability of such accelerometers without the need for hardware solutions, digital signal compensation and error rejection are deployed in software. This method eliminates the need for signal amplifiers and conditioners onboard the sensing nodes.

Simple LSTM networks deployable on edge devices are effective as signal compensators for increasing the usable frequency range of an accelerometer.[4] In the method proposed by Satme et al., the accelerometer for the SHM sensor package is used alongside a higher-quality reference accelerometer to record low-frequency vibrations generated by an electromagnetic shaker. The network is then trained using a supervised learning approach to produce an output similar to the reference accelerometer using only the package accelerometer as input, thereby increasing its usable range. Since this method targets inexpensive, low-power sensor packages, it would greatly benefit from a reduction in the number of computations required to perform inference.

As our goal is to move towards deploying signal compensation models on real-world SHM sensing nodes, we must select a platform to target prior to designing the model. The UAV-deployable SHM sensing node proposed by Yount et al.[12] is uniquely suited for this task. This sensing node, shown in Figure 2, uses the Arm Cortex-M7-based Teensy 4.0, which features a single execution unit running at 600Hz and an FPU. The proposed technique would be ideal for this device, as the presence of an FPU and the lack of significant parallelization features makes its dense-matrix and bit precision preserving properties favorable over quantization and pruning when considering model latency.

(a) (b)

Figure 2. The drone-deployable sensor package with (a) a field deployment; and (b) key components annotated.

3. METHODOLOGY

This section introduces the methodology used in this work.

3.1 Degree-of-Freedom Decomposition

To improve upon low-rank approximation, we propose the use of a degree-of-freedom-based approach as an alternative to using the SVD directly for inference. For a matrix with rank $r < \min(m, n)$, all rows exist in the span of only r 'basis' rows, and the remaining $m - r$ rows may be written as a transform of these 'basis' rows. The row-vector products for the remaining rows can be computed as the transform of the row-vector products of the basis rows, with a saving of $n - r$ multiply-adds for each row. This leads to a two-step process:

$$Ax_1 = Bx, \tag{3}$$

$$Ax_2 = Cx_1, \tag{4}$$

$$Ax = P \begin{bmatrix} Ax_1 \\ Ax_2 \end{bmatrix}. \tag{5}$$

B is an $r \times m$ matrix, C is $(n - r) \times r$, and P is a permutation matrix used to chose the r basis rows. In this form, the matrix-vector multiplication takes

$$mr + (n - r)r = mn - (m - r)(n - r) \text{ multiplies, and} \tag{6}$$

$$(mr - r) + (r(n - r) - (n - r)) = (mn - n) - (m - r)(n - r) \text{ adds.} \tag{7}$$

This provides a significant improvement over the direct use of SVD inference, as shown in Figure 3.

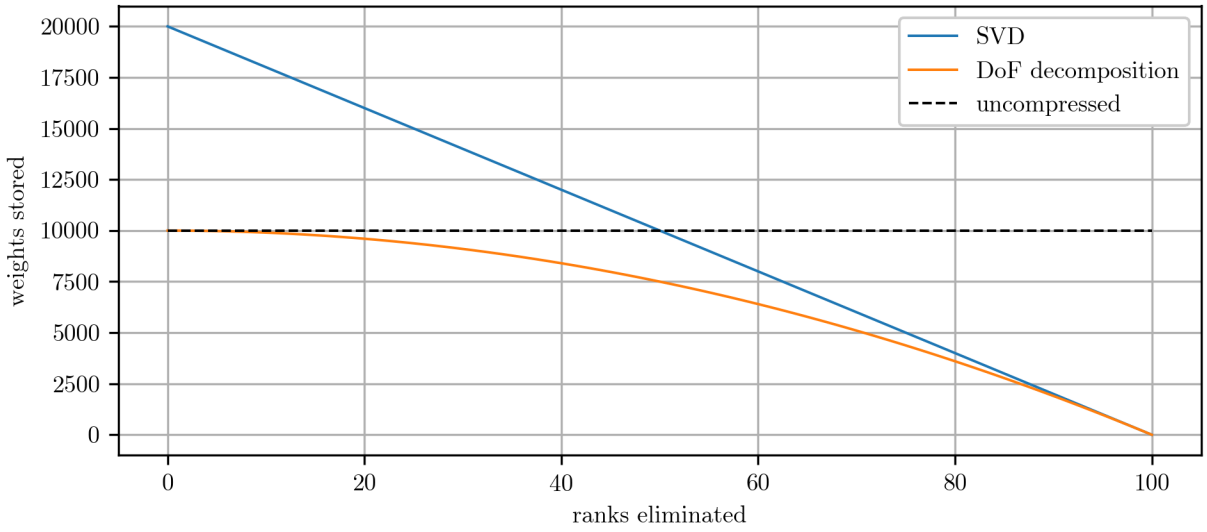

Figure 3. The number of weights being stored using both compression methods on a 100×100 full-rank matrix.

The first terms on the right-hand side show the amount of computations for the unreduced matrix, so we can see that any rank reduction always decreases the amount of computation. We also note that $(m-r)(n-r)$ is a quadratic expression of the number of ranks reduced so that each subsequent rank reduction produces more saved computations than the previous one. From the rank-reduced SVD, we can divide U into an upper square $r \times r$ matrix U_1 and a lower matrix U_2,

$$U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}. \tag{8}$$

We note that there is no guarantee that the submatrix U_1 taken from the first r rows and columns will have an inverse, however after permuting rows, it can be guaranteed either: (1) an invertible submatrix can be found; or (2) U is actually rank $r-1$ or lower, and a greater reduction can be performed. In practice, this is a non-issue since the chance that the pseudo-random weight matrix is non-singular is very low. The equations of the matrices B and C are

$$B = U_1 \Sigma V^{\mathrm{T}}, \tag{9}$$

$$C = U_2 U_1^{-1}. \tag{10}$$

We are assuming here that pre-computing a matrix inverse U_1^{-1} is feasible; a reasonable assumption when working with tiny/edge-deployed ML.

3.2 LSTM

LSTM is a form of a recurrent neural network that employs a gated structure to control the flow of error between states, thus enabling the model to choose to retain or forget information.[13] This process is given by

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{11}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{12}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{13}$$

$$\bar{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \tag{14}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \bar{c}_t, \tag{15}$$

$$h_t = o_t \circ \tanh(c_t). \tag{16}$$

Here, W_f is the weight matrix responsible for managing the forget gate, W_i, the input gate, and W_o, the output gate. W_c is responsible for the creation of the candidate cell state $\bar{c}_t$. Element-wise multiplication is denoted by $\circ$, and σ represents the sigmoid activation function. This process results in the production of the hidden state vector h_t, which provides the prediction of the model.

From the equations, LSTM inference employs four separate weight matrices—each for the input, forget, cell, and output gates—resulting in four distinct matrix-vector multiplications. However, it is common practice for LSTM implementations to consolidate these weight matrices into a single matrix, enabling inference with just one matrix-vector multiply. This results in a streamlined matrix structure that is easier to deploy on edge devices. The consolidated weight matrix also provides a simple way to demonstrate the potential of this technique, as there is only one matrix to determine the low-rank approximation of. For these reasons, we elect to use the following procedure to create the consolidated weight matrix and perform inference in a single matrix-vector multiply.

$$W = \begin{bmatrix} W_i & W_f & W_c & W_o \\ U_i & U_f & U_c & U_o \end{bmatrix}, \tag{17}$$

$$b = \begin{bmatrix} b_i \\ b_f \\ b_c \\ b_o \end{bmatrix}, \tag{18}$$

$$y = \begin{bmatrix} x \\ h \end{bmatrix}, \tag{19}$$

$$\begin{bmatrix} z_i \\ z_f \\ z_c \\ z_o \end{bmatrix} = Wy + b, \tag{20}$$

$$\begin{bmatrix} i_t \\ f_t \\ \bar{c}_t \\ o_t \end{bmatrix} = \begin{bmatrix} \sigma(z_i) \\ \sigma(z_f) \\ \tanh(z_c) \\ \sigma(z_o) \end{bmatrix}. \tag{21}$$

3.3 Training

Following Satme et al., a signal compensation model consisting of a single 50-unit LSTM cell and a dense layer was trained. To compile a training dataset, a sensor package equipped with an SCA3300 accelerometer was magnetically attached to a case containing a MEMS accelerometer, which acts as the reference. This assembly was then placed on an electromagnetic shaker, as shown in Figure 4. The electromagnetic shaker was controlled with a function generator, which enabled the selection of specific excitation frequencies.

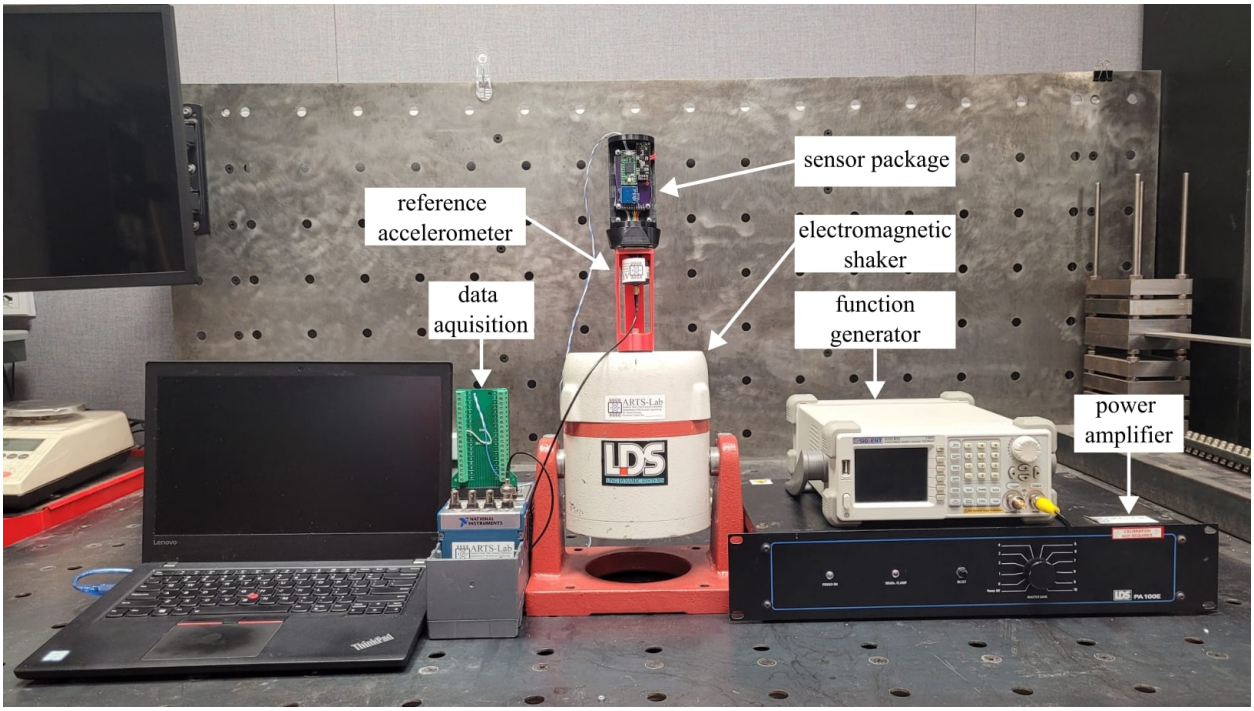

Figure 4. Data collection and experimental setup.

The training dataset consists of frequency sweeps in the range of 1Hz to 10Hz, which enables the model to be exposed to all frequencies within the range of interest. Given the low-frequency vibrations involved in SHM, we determined a target sample rate of 400Hz to be sufficient. The signals recorded by the package and reference accelerometer were aligned using interpolation prior to training. The model was then fitted to the reference data using the package data as input.

3.4 Model Conversion and Validation

Following the truncated SVD approach, the LSTM weight matrix was reduced by computing its SVD, then truncating the last $r - k$ columns in U, rows in V^{T}, and singular values in the diagonal matrix Σ. For our application, we determined a reduction from rank 51 to rank 41 to be sufficient. We then used the above process to generate the B and C matrices required for the DoF decomposition. Since the compensation model consists only of an LSTM and a dense layer, this was the only reduction necessary.

To validate the model, a variation of the TensorFlow LSTM was constructed. This reduced LSTM was modified to replace its forward pass with the DoF decomposition, allowing inference to be performed with the new method. Performance was then evaluated using a validation dataset consisting of a frequency sweep similar to what was used in training.

Table 1. Performance comparison between the original model and the compressed model.

testing	$\mathrm{SNR_{dB}}$	MSE	RMSE	MAE	TRAC	Parameters
uncompressed	14.8498	0.0005	0.0221	0.0168	0.9673	10451
compressed	14.0765	0.0006	0.0242	0.018	0.9616	8861
difference	0.7733	-0.0001	-0.0021	-0.0012	0.0057	1590
% difference	5.3466	17.7586	8.8969	7.1053	0.5925	16.4664

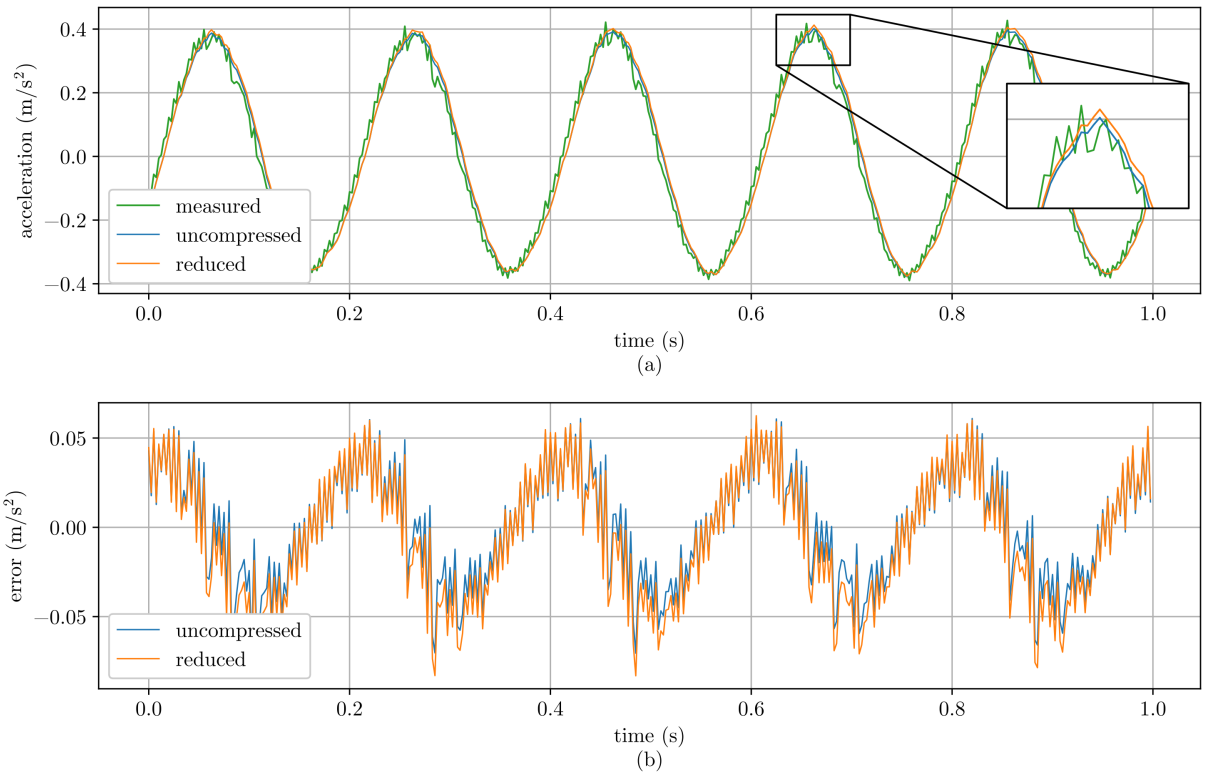

Figure 5. The non-reduced model and its reduced counterpart are being tested on a validation dataset window.

4. RESULTS AND DISCUSSION

To evaluate the reduced model, it was compared against the uncompressed model using SNR, MSE, RMSE, MAE, TRAC, and the number of parameters present in the model as metrics, as shown in Table 1. The DoF decomposition method enabled the memory footprint of the model to be reduced by over 16%; a significant achievement given the number of reductions performed was well below the break-even point required to make the direct use of SVD viable. This reduction in parameters will better facilitate edge deployment, as it not only lowers the model's memory footprint, but also the number of computations required to perform inference.

The circumvention of the break-even point provided by the improved method enabled a conservative approach to rank reduction, leading to only a slight compromise in functionality. The reduced model was shown to only decrease SNR by 5.3466% and increase MSE by 17.7586%, which is in an acceptable range for our purposes. The TRAC for the reduced model remained high, indicating a strong similarity between the reference signal and the output of the reduced model. While we found the performance of the compressed model acceptable, we note that fewer reductions can be done while still providing a gain in space and time complexity should more accuracy be desired.

Based on the above results, we find the reduced model to be fitting for deployment on the Cortex-M7-based UAV-deployable SHM sensing node. The reduced memory footprint enables a larger data buffer, while the reduction in operation count frees up computational resources for other online data analysis techniques that may be deployed alongside the model.

A direct comparison between the outputs of the compressed and uncompressed models is made in Figure 5 (a), where their similarity is made apparent. The similarity between the error shown in 5 implies the reduction process exaggerated existing flaws in the model rather than creating new ones. As with other compression techniques, its efficacy is dependent on the quality of the model prior to compression.

5. CONCLUSION

Low-rank approximation sees little use in comparison to other methods such as quantization and pruning, due in no small part to the high barrier to entry posed by the $\lceil m/2 \rceil$ rank reductions a direct deployment would require. However, the preservation of dense matrices and the high degree of control over the error threshold provided by the technique encourages further development. As an improvement over the direct use of SVD for inference in models compressed with low-rank approximation, a degree-of-freedom-based decomposition of the matrix-vector product was utilized, which yields computational gains at any level of reduction. This technique was applied to a model trained for vibration signal compensation, where it provided a 16.5% reduction in model parameters while only increasing the model's MSE by 17.8%. These results reveal a promising future for low-rank approximation as a model compression technique in the field of edge machine learning. Future work will involve the deployment of this technique on an SHM sensing node and improving the method used to select ranks for elimination.

ACKNOWLEDGMENTS

This material is based upon work supported by the Air Force Office of Scientific Research (AFOSR) through award no. FA9550-21-1-0083. This work is also partly supported by the National Science Foundation (NSF) grant numbers CCF - 1956071, CMMI - 2152896, CCF-2234921 and, CPS - 2237696. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Air Force.

REFERENCES

[1] Hua, H., Li, Y., Wang, T., Dong, N., Li, W., and Cao, J., "Edge computing with artificial intelligence: A machine learning perspective," **55**(9), 1–35.

[2] Immonen, R. and Hämäläinen, T., "Tiny machine learning for resource-constrained microcontrollers," **2022**, 1–11.

[3] Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., and Hafid, A. S., "A comprehensive survey on TinyML," **11**, 96892–96922.

[4] Satme, J. N., Coble, D., Huang, H.-T., Downey, A. R. J., and Bakos, J. D., "Non-linear vibration signal compensation technique for UAV-deployable sensor packages with edge computing," in [*Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2023*], Su, Z., Limongelli, M. P., and Glisic, B., eds., SPIE (apr 2023).

[5] Merenda, M., Porcaro, C., and Iero, D., "Edge machine learning for ai-enabled IoT devices: A review," *Sensors* **20**(9) (2020).

[6] Zhu, M. and Gupta, S., "To prune, or not to prune: exploring the efficacy of pruning for model compression," (2017).

[7] Zhuo, L. and Prasanna, V. K., "Sparse matrix-vector multiplication on FPGAs," in [*Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays*], *FPGA '05*, 63–74, Association for Computing Machinery, New York, NY, USA (2005).

[8] Winata, G. I., Madotto, A., Shin, J., Barezi, E. J., and Fung, P., "On the effectiveness of low-rank matrix factorization for LSTM model compression," (2019).

[9] Dantas, P. V., Sabino da Silva, W., Cordeiro, L. C., and Carvalho, C. B., "A comprehensive review of model compression techniques in machine learning," **54**(22), 11804–11844.

[10] Zhu, L., Fu, Y., Chow, R., Spencer, B., Park, J., and Mechitov, K., "Development of a high-sensitivity wireless accelerometer for structural health monitoring," **18**(1), 262.

[11] Kavitha, S., Joseph Daniel, R., and Sumangala, K., "Design and analysis of MEMS comb drive capacitive accelerometer for SHM and seismic applications," *Measurement* **93**, 327–339 (2016).

[12] Yount, R., Satme, J. N., and Downey, A. R. J., [*Frequency-Based Damage Detection Using Drone-deployable Sensor Package with Edge Computing*], 67–73, Springer Nature Switzerland (Aug. 2024).

[13] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," **9**(8), 1735–1780.