Optimized Deep Siamese Network for Structural Health Monitoring Using Vibration-Based Measurements

MANASHI SAHARIA, MOUMITA ROY
and NIRMALENDU DEBNATH

ABSTRACT

Structural health monitoring (SHM) is essential for evaluating infrastructure conditions and ensuring its safety. Over time, the structural integrity of buildings and civil infrastructure can deteriorate, leading to a decline in performance and reliability. SHM has increasingly used deep learning (DL) models for effective damage detection. However, for onsite damage detection of smart structures, there is a need to deploy these trained DL models on resource-constrained devices such as Raspberry Pi, Arduino, mobile devices, or microcontrollers. To accomplish this task, the trained DL models need to be optimized for deployment that enables efficient and timely damage detection in real-world applications by testing the vibration (acceleration) measurements coming from the sensors placed over the structures. As per the knowledge of the authors, this is a crucial aspect that has yet to be fully explored in SHM. In the present work, vibration signals are initially transformed into a 2D matrix format. These transformed 2D matrices are used as input to a DL model, specifically the deep-siamese network (DSN). The DSN operates based on similarity measures between pairs of training samples. This enables the proposed SHM approach to detect and report previously unseen damage scenarios during testing. To enable real-time inference, the DSN model is optimized using pruning, quantization, and knowledge distillation. These techniques reduce the size of the model and the inference time, making it suitable for deployment on resource-constrained devices. To validate this approach, the IASC-ASCE benchmark structure with varying damage conditions is used. The results show that the optimized model maintains high detection accuracy while enabling fast and efficient decision making. Each optimization technique is used to convert the DSN model into a lightweight (.tflite) format. The model is then deployed on a virtual machine running Raspberry Pi OS with Debian Linux. These optimizations support real-time structural health monitoring with high detection accuracy. The model also handles unseen damage during testing, making it suitable for real-world deployment. This makes it a promising solution for real-world SHM, bridging the gap between DL advancements and practical online monitoring.

Manashi Saharia, msaharia02@gmail.com, Dept. of CSE, IIIT Guwahati, Assam, India
Moumita Roy, moumita2009.roy@gmail.com, Dept. of CSE, IIIT Guwahati, Assam, India
Nirmalendu Debnath, nirmalendu.debnath@gmail.com, Dept. of Civil Engg., NIT Silchar, Assam, India (corresponding author)

INTRODUCTION

Structural health monitoring (SHM) is a method used to assess the condition of various civil/engineering structures over time to improve their safety [1]. It is an interdisciplinary field that integrates the principles of engineering, data science, and sensor technology in a single platform. It helps to detect damage and predict failures in bridges, buildings, aircraft, etc. SHM involves collecting and processing vibration, strain, and other signals to detect damage and ensure structural reliability. It improves maintenance efficiency by allowing for early fault detection and real-time monitoring. The main objective is to automatically identify damage and maintain structural integrity for the safety of both people and property. Traditional damage detection methods, such as visual inspection and non-destructive techniques (NDT), are often impractical for large structures due to high costs and inefficiency. To address these challenges, automated real-time structural damage detection (SDD) methods have emerged, including global vibration-based and local assessment approaches [1]. With advancements in sensors and data processing, data-driven techniques like machine learning (ML) and deep learning (DL) offer faster, more efficient alternatives to physics-based models. Unlike ML, which requires manual feature extraction, DL models automatically learn features from raw vibration signals. Among DL models, convolutional neural networks (CNNs) are widely used for fault detection. While 1D CNNs are common, 2D CNNs capture richer features and improve accuracy. In a study, Li et al. [2] proposed a 1D CNN approach for bridge damage detection using vibration data. With increasing infrastructural complexity, real-time SHM has become essential for early damage detection. Deploying DL models on resource-constrained edge devices such as Raspberry Pi, Arduino, and microcontrollers enables efficient on-site data processing without relying on large computing systems [3, 4]. However, due to limited memory and processing power, DL models trained on high-performance local machines must be further optimized before deployment. Techniques such as quantization, pruning, and knowledge distillation help to reduce model size and increase inference speed by maintaining its accuracy [5]. Although still emerging in SHM, optimized DL models have shown success in other fields. Stephany et al. [6] used CNN and SHAP-based feature selection for canola yield prediction on edge devices. Lee et al. [7] optimized lightweight convolution models for efficient image classification.

In recent years, DL algorithms, particularly CNN, are widely used in SHM. However, deep siamese networks (DSNs) remain largely unexplored despite their advantage of learning similarity between sample pairs. Unlike CNNs, DSN require less training data and can report the unseen damage scenarios during testing. The purpose of the present work is two-folded: (a) global-level damage detection using vibration-based measurements, and (b) deployment of the optimized DL model on a resource-constrained device for real-time SHM. The present work proposes a framework that transforms vibration signals into a matrix representation and uses DSN to learn discriminative features, as it is based on similarity measures. To enable real-time inference on a resource-constrained device, the DSN is optimized using techniques such as knowledge distillation, pruning, and post-training quantization. These optimizations significantly reduce model size and computation time. The subsequent sections describe the proposed methodology, the experimental setup, and the performance obtained from the optimized framework.

PROPOSED METHODOLOGY

As already discussed, the DSN model accepts input in 2D form to extract discriminative features. In this regard, this work presents a methodology that converts vibration signals into matrix form and uses them as input to a DSN. After optimization, the DSN model is deployed on a resource-constrained device (i.e., virtual machine running Raspberry Pi OS with Debian Linux) to enable efficient real-time inference for SHM.

Deep siamese network

A novel DSN-based damage detection approach is proposed to address CNN limitations [8]. The DSN consists of two identical subnetworks that share weights and act as a binary classifier. Each extracts features from input pairs and measures their difference. The network minimizes distances for similar pairs and maximizes them for dissimilar pairs through a gradient descent learning algorithm. In this study, initially, 1D vibration signals are reshaped into 2D matrices to capture spatial and temporal features [9]. Subsequently, paired training samples are created with the label $Y_s = 1$ for positive pairs and $Y_s = 0$ for negative pairs. Each CNN branch extracts feature vectors $\vec{f}_{x_i}$ and $\vec{f}_{x_j}$, and their Euclidean distance d_s is calculated as follows:

$$d_s = ||\vec{f}_{x_i} - \vec{f}_{x_j}||^2 \tag{1}$$

The contrastive loss $L_{\text{cons\_loss}}$ is defined as:

$$L_{\text{cons\_loss}} = \frac{1}{N}\sum_{s=1}^{N}\left[(1-Y_s)\frac{1}{2}(d_s)^2 + Y_s\frac{1}{2}\max(l-d_s,0)^2\right] \tag{2}$$

The DSN minimizes d_s for similar pairs and maximizes it for dissimilar pairs using a margin l, trained via gradient descent. During testing, each test sample is paired with representative samples (mean feature vectors from the training set) and passed through the DSN. If the similarity score exceeds 0.5, the sample is classified as positive and assigned the corresponding class label; otherwise, it is classified as negative. For a positive pair, the class label of the test sample is assigned the same as the class label of the corresponding representative training sample.

Resource-constrained device

As already discussed, with increasing structural complexity, the need for real-time SHM has become critical for timely damage identification. In this regard, ML/DL models must be adapted to deploy on resource-constrained edge devices such as Raspberry Pi, Arduino, and microcontrollers that facilitate efficient on-site data processing without relying on a high-performance computing setup. Resource-constrained devices are hardware components that perform data collection, processing, and analysis locally, close to the source of data generation. Unlike cloud computing, which requires transmitting data to centralized servers, edge computing processes data on the device or near the node, reducing transmission time and bandwidth usage, thus improving speed and efficiency [4]. In this study, the optimized DSN model is deployed on a Raspberry Pi virtual machine

running Debian Linux, offering a lightweight and flexible environment ideal for real-time applications such as SHM and industrial automation. By performing computations locally, resource-constrained devices minimize latency and improve system reliability. Moreover, it helps to maintain data privacy, as sensitive information does not need to be sent to external servers for processing.

Lightweight models for implementation on resource-constrained device

Real-time deployment of DL models on resource-constrained devices requires careful optimization due to their limited memory and processing power. Techniques such as quantization, pruning, and knowledge distillation are employed to reduce the size of the model and enhance the speed of inference by maintaining accuracy. In this study, the DL model (DSN) is first trained on a local machine and then further optimized to create a lightweight version, which is suitable for deployment on edge platforms [4]. These methods address the limitations of traditional DL models and make them suitable for on-site processing in SHM. The optimization techniques utilized in the proposed framework are discussed in the following sections.

KNOWLWDGE DISTILLATION

Knowledge distillation (KD) is a model compression technique in which a smaller student model learns from a larger teacher model by mimicking its soft predictions [10]. Here, in the present study, after obtaining the best trained DSN model (used as a teacher model) using hyperparameter tuning and k-fold cross validation, KD is applied to achieve the student model. This is obtained by optimizing a combined loss function:

$$L = (1 - \alpha)L_{\text{CE}} + \alpha L_{\text{KD}} \tag{3}$$

where L_{CE} is the standard cross-entropy loss between the student model's predictions and ground truth labels, and L_{KD} represents the distillation loss, typically computed as the Kullback-Leibler (KL) divergence between the soft probability outputs (logits) of the teacher and student models. The parameter α controls the trade-off between direct supervision from ground truth labels and knowledge transfer from the teacher. By leveraging knowledge distillation, the student model captures the essential knowledge of the teacher while significantly reducing computational complexity.

PRUNNING

After the student model has acquired essential knowledge from the teacher via knowledge distillation, pruning is applied to further compress the model [10]. As the student already encapsulates the teacher's knowledge, pruning removes redundant parameters while preserving accuracy [4]. This results in a more lightweight and efficient model, well-suited for deployment on resource-constrained devices.

POST-TRAINING QUANTIZATION

Post-Training Quantization (PTQ) is an optimization technique applied after training, converting model weights and activations from 32-bit floating-point (FP32) to lower-

precision formats like INT8 [4, 10]. It reduces model size and improves inference speed without retraining. The quantization of a value x can be expressed as:

$$x_{\text{quantized}} = \text{round}\left(\frac{x - \min(x)}{\text{scale}}\right) \tag{4}$$

where $\min(x)$ is the minimum value in the data range, and scale maps FP32 to INT8. In the current implementation, after applying pruning, PTQ is then applied to further optimize the model. This step directly converts the model into a lightweight format, such as a TensorFlow Lite (.tflite) model. This conversion ensures that the model is highly optimized for deployment on resource-constrained devices where memory and computation power are critical. This results in a smaller, faster, and more efficient model. This quantized .tflite model is made ready to deploy ina Raspberry Pi virtual machine (VM) for real-time inference.

The graphical abstract of the overall proposed framework is illustrated in Figure 1.

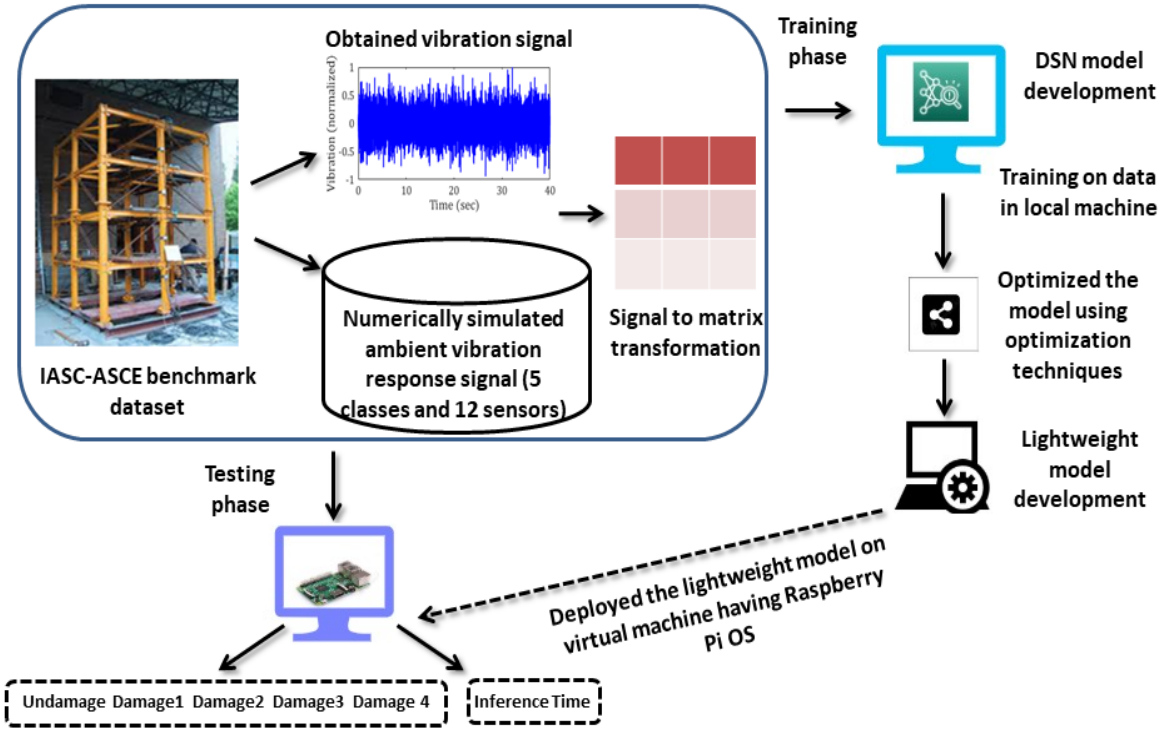

Figure 1. Graphical representation of the proposed method.

DATASET DESCRIPTION

The proposed DL approach is validated using a modified IASC-ASCE steel frame dataset (as shown in Figure 1). Twelve global DOFs are selected from vibration responses of a 120 DOF numerical model, transformed from local to global coordinates. The dataset includes five health states, with acceleration responses sampled at 200 Hz. A total of 5000 samples (1000 per health state) are used, each containing vibration signals of 8000 timestamps from 12 sensors. However, the first 1000 timestamp signals

are removed to avoid noise. The structural modifications include: The changes from the original structure are:

a) No stiffness in the braces of the first story.

b) No stiffness in the braces of the first and third stories.

c) No stiffness in one brace in the (+x) face of the first story.

d) No stiffness in one brace in the (+x) face of the first story and (-y) face of the third story.

RESULT AND DISCUSSION

To evaluate the proposed method, experiments have been conducted using the IASC-ASCE benchmark dataset with five classes. A DSN model has been used with transformed vibration signals as input. The main focus is on optimizing the DL model for lightweight deployment on resource-constrained devices. Evaluation included metrics such as accuracy, precision, recall, F1-score, model size, and inference time.

Details of the experimentation

As mentioned, the DSN architecture addresses challenges faced by CNN-based global damage detection, such as identifying unknown damage scenarios during testing. To validate this, experiments were conducted in two setups:

1) Setup 1: Experimentation using all available samples under k-fold cross-validation.

2) Setup 2: Experimentation with unseen damage scenarios (having the samples from the damage classes not present during training).

The IASC-ASCE benchmark dataset has been used for this implementation. The vibration signals have been reshaped into 2D matrices ($70 \times 100 \times 12$), which are normalized using min-max scaling, and fed into the DSN model. The model has been trained with various hyperparameter settings and tuned through grid search. The best model has been identified using 3-fold cross-validation and tested on a separate 20% test dataset. For real-time inference, the model has been optimized using knowledge distillation, pruning, and post-training quantization to a lightweight (.tflite) format. Performance and inference time are compared at each stage, and the optimized model has been deployed on a Raspberry Pi VM running Debian Linux for online evaluation.

RESULT ANALYSIS UNDER SETUP 1

In Table I, the best-trained DSN model (learning rate 0.001, 20 epochs, batch size 32) achieved 79.5% accuracy, improving to 79.7% after KD and maintaining performance after pruning. PTQ (TFLite format) boosted accuracy to 80.5%. Class-wise accuracies in Table II showed improvement in Damage Class 1 (D1) with knowledge distillation, while pruning maintained the performance. Quantization enhanced the accuracy for Damage Classes 3 and 4 (D3, D4). Further, in Table III, the original DSN model had the largest size (1.62 MB) and slowest inference (254.72 seconds). KD reduced the size to 84.4 KB, and pruning increased the size to 157 KB but maintained the speed. Further, quantization reduced the size to 28.8 KB and inference time to 92.82 seconds. On the Raspberry Pi VM, the quantized model achieved 32.9 seconds, showing high efficiency for real-time deployment.

TABLE I. COMPARATIVE RESULTS OBTAINED USING THE IASC-ASCE BENCHMARK DATASET UNDER SETUP 1

Model	**Accuracy (%)**	**Precision**	**Recall**	**F1 score**
DSN	79.5	0.632	0.795	0.704
DSN+ KD	79.7	0.636	0.797	0.707
DSN + KD + Pruning	79.5	0.636	0.795	0.704
DSN + KD + Pruning + PTQ	**80.5**	**0.649**	**0.805**	**0.718**

TABLE II. CLASS-WISE ANALYSIS OF THE PROPOSED APPROACH UNDER SETUP 1

Model	**Undamge**	**D1**	**D2**	**D3**	**D4**
DSN	80.0	77.9	80.1	79.2	80.3
DSN + KD	78.1	**81.0**	80.2	79.3	80.3
DSN + KD + Pruning	**80.1**	78.9	79.0	78.9	80.3
DSN + KD + Pruning + PTQ	79.7	80.1	**80.8**	**80.7**	**81.4**

TABLE III. INFERENCE TIME AND MODEL SIZE UNDER SETUP 1

Category	**Model**	**Model Size**	**Inference Time (secs)**
	DSN	1.62 MB	254.72
On local machine	DSN + KD	84.4 KB	242.57
	DSN+ KD + Pruning	157 KB	249.69
	DSN + KD + Pruning + PTQ	28.8 KB	92.82
On VM (Raspberry Pi)	Quantized (.tflite) model	**28.8** KB	**32.90**

RESULT ANALYSIS UNDER SETUP 2

For the implementation of Setup 2, one class (damage class 4) is treated as unseen and reserved exclusively for testing. At the same time, training is carried out using samples from the remaining classes i.e., undamage class and damage classes 1, 2, and 3. In the split test data, the unseen damage class (damage class 4) has been concatenated, and further testing is performed by pairing these unseen class samples with representative samples from the known classes. Finally, the percentage of correctly classified samples from the unseen class is reported to evaluate the model. From Table IV, it can be analyzed that the results obtained using the DSN model achieved 88.8% accuracy on a local machine. Further, with optimizations like knowledge distillation, pruning, and PTQ, the model size and inference time are reduced significantly. It can be observed that the quantized DSN model achieved 88.3% accuracy and a remarkably faster inference time of 121.80 seconds on a Raspberry Pi VM.

TABLE IV. PERFORMANCE OBTAINED UNDER THE SETUP 2

Category	**Model**	**Percentage**	**Model Size**	**Inference Time (secs)**
	DSN	88.8	1.62 MB	1539.07
On local machine	DSN + KD	88.8	84.4 KB	1394.85
	DSN + KD + Pruning	**89.0**	157 KB	1501.01
	DSN + KD + Pruning+ PTQ	88.3	28.5 KB	135.98
On VM (Raspberry Pi)	Quantized (.tflite) model	88.3	**28.5** KB	**121.80**

CONCLUSION

This study demonstrated that a DSN model can effectively detect structural damage by transforming vibration signals into 2D matrices and deploying the model on resource-constrained devices. Using knowledge distillation, pruning, and quantization, the model size has been reduced from 1.62 MB to 28.5 KB, and inference time from over 1500 seconds to under 33 seconds, without much variation in accuracy. The model is also enable to efficiently report the presence of unseen damage classes during real-time monitoring of the operational structure. In the future, the investigation may be conducted with various resource-constrained devices for diverse types of structures in a multimodal DL framework.

ACKNOWLEDGMENT

The authors acknowledge the support from the Science and Engineering Research Board (SERB), Department of Science and Technology (DST), Government of India (project grant with file No. CRG/2021/007352).

REFERENCES

1. Avci, O., O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj, and D. J. Inman. 2021. "A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications," *Mechanical systems and signal processing*, 147:107077.
2. Li, S., X. Zuo, Z. Li, and H. Wang. 2020. "Applying deep learning to continuous bridge deflection detected by fiber optic gyroscope for damage detection," *Sensors*, 20(3):911.
3. Dang, H. V., M. Tatipamula, and H. X. Nguyen. 2021. "Cloud-based digital twinning for structural health monitoring using deep learning," *IEEE transactions on industrial informatics*, 18(6):3820–3830.
4. Ameen, S., K. Siriwardana, and T. Theodoridis. 2023. "Optimizing Deep Learning Models For Raspberry Pi," *arXiv preprint arXiv:2304.13039*.
5. Schizas, N., A. Karras, C. Karras, and S. Sioutas. 2022. "TinyML for ultra-low power AI and large scale IoT deployments: A systematic review," *Future Internet*, 14(12):363.
6. Valarezo-Plaza, S., J. Torres-Tello, K. D. Singh, S. J. Shirtliffe, S. Deivalakshmi, and S.-B. Ko. 2024. "A Novel Optimized Deep Learning Model for Canola Crop Yield Prediction on Edge Devices," *IEEE Transactions on AgriFood Electronics*, 2(2):436–444.
7. Lee, H., N. Lee, and S. Lee. 2022. "A method of deep learning model optimization for image classification on edge device," *Sensors*, 22(19):7344.
8. Kalita, I. and M. Roy. 2022. "Class-wise subspace alignment-based unsupervised adaptive land cover classification in scene-level using deep siamese network," *IEEE Transactions on Neural Networks and Learning Systems*, 34(7):3323–3334.
9. Khodabandehlou, H., G. Pekcan, and M. S. Fadali. 2019. "Vibration-based structural condition assessment using convolution neural networks," *Structural Control and Health Monitoring*, 26(2):e2308.
10. Kim, J., S. Chang, and N. Kwak. 2021. "PQK: model compression via pruning, quantization, and knowledge distillation," *arXiv preprint arXiv:2106.14681*.