IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 February 2024, accepted 8 February 2024, date of publication 13 February 2024, date of current version 22 February 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3365550*

APPLIED RESEARCH

# An Investigation Into the Advancements of Edge-AI Capabilities for Structural Health Monitoring

**ANOOP MISHRA, GOPINATH GANGISETTI, AND DEEPAK KHAZANCHI**
College of Information, Science and Technology, University of Nebraska, Omaha, NE 68182, USA

Corresponding author: Anoop Mishra (amishra@unomaha.edu)

This work was supported in part by the U.S. Army Corps of Engineers, Engineering Research and Development Center (ERDC), under Contract W912HZ21C0060 and Contract W912HZ23C0005.

**ABSTRACT** The aim of this study is to investigate the capabilities of edge-AI in the field of structural health monitoring, with a particular emphasis on detecting cracks in concrete bridges. Comprehensive literature suggests that edge-AI approaches have not been utilized in the structural health monitoring domain (SHM). This article proposed two novel frameworks: an edge-AI framework for the SHM domain and a cloud-edge adaptive intelligence for crack detection (CEAIC) to utilize edge-AI approaches for real-time scenarios. The framework incorporates a novel edge-AI framework, the crowd intelligence approach, and the CrANET framework to perform weakly supervised crack segmentation. Quantization approaches are used to transform the deep learning model into an optimized model to be compatible with edge devices. The edge-AI experiments for the cracks detection task are conducted using Kneron KL520 and Google Coral development board. A responsive website has been developed to demonstrate the real-world implications of the CEAIC framework. This study has the potential to provide a cost-effective and reliable solution for real-time monitoring and assessment of concrete bridge cracks, thereby improving the safety and longevity of bridges. The outcomes of this research endeavor will furnish us with invaluable insights regarding the feasibility and potential advantages of incorporating edge AI into the SHM domain.

**INDEX TERMS** Cracks detection, edge-AI, neural networks, quantization, structural health monitoring.

## I. INTRODUCTION

According to the latest National Bridge Inventory (NBI) data, as of 2021, over 4 out of 10 bridges in the United States were either structurally deficient or functionally obsolete. The American Society of Civil Engineers (ASCE) infrastructure report (https://infrastructurereportcard.org/cat-item/bridges/) indicates 7.5% out of 600,000 of the nation's bridges are labeled as ''poor'' in condition. The failure of these bridges can have a direct impact on public safety and economic growth; for example, they are major assets of transportation infrastructure and help to serve communities [6], [36].

However, the increasing traffic loads and the threat of structural fatigue make it crucial to monitor the condition of bridges. Structural Health Monitoring (SHM) can play a key role in increasing the lifespan and performance of bridges [4]. The optimal system for monitoring, managing, and assessing the condition of bridges is required for performing tasks such as bridge maintenance and deterioration. Periodic and continuous monitoring via bridge deck inspections is necessary for structural health monitoring (SHM) [19], [63]. Advances in technology like wireless sensor networks and artificial intelligence have allowed bridge structural health monitoring to be more effective. In the past, AI approaches including machine learning, computer vision, and robotics along with data mining and image processing techniques are utilized to address the SHM challenges [19], [22], [25], [45], [62], [63], [66]. Especially deep learning approaches are broadly utilized in the past for detecting and segmenting bridge components [21], [27], [42]. Results obtained through these computational approaches have shown immense potential to support health monitoring tasks such as crack detection,

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

segmentation of bridge components, UAV inspection of bridges, and damage detection for bridge structure [22], [25], [60], [62]. However, real-time inferences in Structural Health Monitoring (SHM) systems pose a significant challenge due to a range of factors, including the complexity of deep learning models that require substantial computational resources, limited computing power, and the need for high accuracy, which is difficult to achieve due to noisy data [65]. Additionally, given the network constraints in real-time settings, efficient data processing can be further impeded. It is important to deploy complex AI models with limited resources by developing lightweight AI models, efficient AI algorithms, and hardware acceleration techniques to address real-time challenges. This article proposes an integration of edge-AI platforms and their approaches in structural health monitoring to optimize the deployment of complex AI models with limited resources.

Edge-AI allows inference and computation on-device rather than on cloud servers or network connections that optimizes the response time and accelerates AI computation when data is generated and inferred [32]. Edge-AI can be utilized as embedded technology that includes machine learning, neural network processing, and run-time mechanisms on the device itself. This leads to the advantages of optimized response time, high privacy, more robustness, better for dedicated AI tasks, and there is no need for internet connectivity everywhere. In a case where data is generated or collected and simultaneously needs to be inferred, edge-AI has been widely recognized as a promising solution because the inference will be done on the device itself. Edge computing is proposed to address this challenge by reducing the computing capability from the cloud data center to the edge side, i.e., data generation source, which enables AI processing with high performance [32]. Zhou et al. [32] discuss that in August 2018, edge-AI emerged in the Gartner Hype Cycle for the first time. In the industry, many companies like Google, Qualcomm, and others are providing edge AI platforms that enable end devices to run ML inferences with pre-trained models locally. Various dedicated edge AI chips are commercially available on the market, like Google Edge TPU, Intel Nervana NNP, Kneron edge-AI chip, and Huawei Ascend 910 and Ascend 310. Mendez et al. [31] and Zhou et al. [32] explain that integrating edge-AI with cloud computing can enhance personalized AI. They claim this by showing levels of edge intelligence based on data offloading reduction. Zhou et al. [32] also claimed that there is no best level of edge intelligence in general, considering the cost of computational latency and energy consumption due to data offloading. They further claimed that edge intelligence is application-dependent and must be determined based on scenarios. The present article acknowledges the aforementioned claim and expounds upon the practical implication of utilizing cloud-edge intelligence. The details of this discussion as an application in the SHM domain are elaborated in section III-D.

### A. RESEARCH GOALS AND PROPOSED APPROACH

In this research study, drawing from artificial intelligence (AI), neural network process acceleration, and edge computing literature, the study aims to investigate edge-AI integration in structural health monitoring tasks to tackle the issues in real time. The objective is to develop deep learning-based neural network models that are optimized for real-time inference at physical bridge sites while balancing accuracy and minimizing latency. A novel edge-AI framework is proposed that integrates edge-AI approaches in SHM tasks. This framework represents a significant advancement in the field of SHM by leveraging the capabilities of deep learning and edge-AI platforms. The edge-AI framework showcases the effectiveness of developing optimized and compatible deep learning models using pruning, quantization, and weight clustering techniques or through on-device training with transfer learning to refine an existing edge neural network model. The framework offers insight into the functionality and application of these approaches and highlights their potential value in tasks like crack detection, rust detection, displacement, vibration monitoring, etc., within the SHM domain. The edge-AI platforms compromising Kneron KL520 and Google Coral board are used to conduct the experiments. Fig. 5 demonstrates the proposed edge-AI framework. A series of experiments is conducted following the framework to perform real-time crack image classification tasks. The details of these experiments are discussed in section III-C4.

Zhou et al. [32] and Mendez et al. [31] discuss integrating cloud computing and claiming that edge intelligence is application-dependent. Acknowledging this discussion and the limitations of edge-AI platforms used for experiments, this study also introduces a novel cloud-edge adaptive intelligence framework for crack detection and quantification (CEAIC) within the SHM domain to address the inherent challenges of performing inference tasks at the physical concrete bridge site. This framework comprises four essential components: proposed edge-AI framework in the SHM domain, cloud computing services, crowd intelligence for gathering human feedback, and adaptive AI for continuous model refinement with new data. Fig. 13 illustrates the CEAIC framework. The CrANET framework, proposed by Mishra et al. [68], is utilized in the proposed CEAIC framework to accomplish the crack detection task. The edge-AI platform compromising Kneron KL520 and Google Coral board performs the binary image classification task, whereas the segmentation, quantification, human feedback recording, and adaptive AI tasks are performed on the cloud using Amazon AWS services. A responsive web application is developed that executes the aforementioned task in the CEAIC framework. This web application is tested on a physical bridge site in real time, validating the proposed CEAIC framework's effectiveness and robust implementation. More details of the experiments can be found in section III.

The use cases, methods, results, and discussion in this study will help and motivate researchers to utilize edge AI for real-time applications in fields such as robotics, smart surveillance, and automation in structural health monitoring. In summary, the technical contributions are as follows:

1) Proposed a novel edge-AI framework for the SHM domain
2) Proposed a novel cloud-edge adaptive intelligence framework for crack detection and quantification (CEAIC) framework in the SHM domain in real-time
3) Developed and deployed the optimized and compatible edge-AI CNN model using attention networks for classifying cracks in image data
4) Integrated crowd intelligence and adaptive AI approach that retrains the model incorporating human feedback

## II. RELATED WORKS

### A. AI/ML AND STRUCTURAL HEALTH MONITORING

In the past, researchers utilized AI/ML approaches such as data mining, image processing, machine learning, computer vision, and robotics to approach the challenges associated with SHM [19], [22], [25], [45], [62], [63], [66]. Gandhi et al. [62] proposed big data, data management, and decision support systems in structure health monitoring. Here, they advocated developing a smart big data pipeline targeting structural health monitoring systems for bridges such that these approaches can reduce manpower requirements, especially for scenarios like manual visual inspection. Many SHM tasks like crack detection, bolts detection, rivets detection, bridge vibration analysis, etc., were approached using deep learning and computer vision [21], [27], [42]. Most of these approaches utilized convolutional neural networks [5], [21], [22], [27], [42]. Segmentation algorithms like Mask RCNN, pyramid networks, and Fast RCNN are used to detect bridge components in SHM [21].

Crack detection or segmentation is an important aspect of bridge health inspection [14], [22], [25]. The inspection attributes include patterns, location, and width of cracks on structural surfaces that help to identify the structural defects and their progression for deriving necessary recommendations for decision-making of maintenance and deterioration [13]. However, most of the crack detection in the literature are designed with high resource requirements, inherit weight redundancy, and include large-scale parameters that lead to a high amount of multiply-accumulate (MAC) operations. For example, ResNets is a 152-layer model that helps to achieve state-of-the-art performance in various computer vision applications. However, increasing the number of layers makes the model resource-hungry. Choudhary et al. [1] explain that VGG16, a convolutional neural network has 138 million learnable parameters and 15.5 billion MACs to perform while inferring a single image. These large-scale parameters and high MAC operations cause high latency in inference time for a given input. In real-time environments, this will be a potential reason for adding loss to the business objectives. Also, this high resource setup of deep neural networks has memory and high computation constraints to deploy on devices such as mobile phones and wearable devices.

Integrating edge-AI approaches can be a feasible solution for deep neural networks to run on mobile devices to provide low latency and high prediction performance. Thus, this paper aims to develop a crack detection model using an image dataset and deploy it on edge devices to optimize the inference time.

### B. EDGE-AI: AN APPLICATION

Edge-AI is an emerging area, and numerous related questions require thoughtful consideration regarding its application, capability, model development, and real-time usage. Recently, Singh and Gill [70] shed light on edge computing paradigms, particularly focusing on Edge AI. It delves into various stages of development for cloudlets, fog computing (FC), mobile edge computing (MEC), and micro-data centers (mDCs). Furthermore, it showcases application use cases for both edge computing and edge-AI that are currently being explored in proof-of-concept studies. Additionally, it analyzes initial commercial service offerings and models. It also conducts a comparative analysis of Edge AI with cloudlets, FC, MEC, and mDCs, considering basic computing characteristics alongside application, functionality, and technological viewpoints. By focusing on the edge-AI approach for deploying AI algorithms and models on resource-constrained edge devices, the survey offers valuable insights. Lastly, to provide direction for future researchers, it highlights potential research areas and presents open challenges. A potential key approach they endorse is edge-to-cloud integration, where Edge AI and cloud computing will be utilized as a hybrid solution. They discussed that the cloud can be used for training and developing ML models, while the edge-AI platforms can perform real-time inference.

Murshed et al. [46] provides a comprehensive overview of edge intelligence with different techniques employed and various applications of edge AI and discusses state of the art. Zhou et al. [32] comprehensively survey the different architectures of edge and artificial intelligence techniques like machine learning, deep learning for prediction, and inference. They discuss that edge-AI emerged in the Gartner Hype Cycle for the first time in August 2018. Zhao et al. [24] proposed ZOO that supports model construction, design, and model uploading on the edge device, including a use-case utilizing the Inception model on Raspberry Pi. Hsieh et al. [37] developed GAIA, which stands for a geo-distributed machine learning system having WAN bandwidth usage to eliminate unnecessary communication between data centers. Zeng et al. [16] proposed a Boomerang framework that performs on-demand cooperative DNN inferencing. Kang et al. [50] developed a cloud model-based scheduler called Neurosurgeon, which partitions the DNN computations between the edge devices and data centers using automation. They discuss that edge-cloud mode is

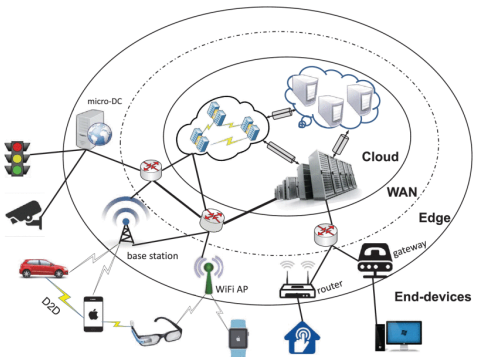



**FIGURE 1.** Edge-Computing use-case illustration by Zhou et al. [32].

similar to edge device mode, and data generated is stored on the cloud [50]. These articles have only high-level discussions. Use cases are merely provided. Contents like experimental challenges, hardware constraints, and limitations are missing.

Badidi et al. [71] proposed a study in the form of a systematic review of Edge AI-assisted video analytics in smart cities by examining ML models and privacy-preserving techniques for edge video analytics. They discuss that edge video analytics, powered by Edge AI, presents exciting possibilities but faces several challenges. The key hurdles include real-time processing, limited resources, power, privacy & security, scalability, robustness & adaptability, and the human-in-the-Loop. They further added that research areas like explainable AI, multi-modal fusion, adversarial robustness, and edge-cloud collaboration hold promise for advancing edge video analytics. While challenges remain, edge video analytics driven by Edge AI has immense potential to revolutionize video analysis and decision-making. They were optimistic about the hurdles and concluded that continuous innovation and adaptation are key to unlocking its full potential.

Wulfert et al. [72] introduce AIfES, a next-generation framework specifically designed for resource-constrained embedded devices. They claim that traditional edge AI frameworks struggle with hardware flexibility and custom hardware integration, limiting their ability to handle modern machine learning advancements. They propose that AIfES offers several advantages, including flexibility on hardware support, on-device training, and modular architecture, and benchmarks show AIfES outperforms TensorFlow Lite for Microcontrollers (TFLM) in both execution time and memory consumption for certain neural network architectures. It even demonstrates the feasibility of training complex CNNs on devices with limited resources. They also aim to expand AIfES by supporting new ANN architectures and further developing federated and online learning techniques. Mendez et al. [31] reveal the edge-AI application in image recognition, like the Caffe2GO framework, which integrates CNN models in mobile devices to accelerate inferencing and transmitting the data to the cloud. Liu et al. [11] built a food recognition system using DNN on edge. Tuli et al. [33] accomplished object detection via embedding YOLOv3 architecture using the COCO dataset on Raspberry PI devices. Lin et al. [34] developed EdgeSpeechNet, a human speech recognition model implemented for mobile phones using edge-AI. Chen et al. [59] implement a small footprint model for edge devices using a deep KWS framework. Zhang et al. [52] utilized edge-AI for autonomous vehicles to process real-time AI tasks like pedestrian detection, object detection, and traffic detection using Intel Movidius. Xu et al. [61] designed STTR, a smart surveillance system to track vehicles in real-time, integrating a camera device on edge.

Based on the discussed literature, edge-AI can be advantageous in the following ways:

1) **Reduce the need for data storage and bandwidth:** By processing data at the edge, edge AI reduces the need to store and transmit data to the cloud or a data center. This can save on data storage and bandwidth costs.
2) **Increase privacy and security:** By processing data locally, edge AI can help to increase privacy and security. Data can be stored locally on devices rather than being transmitted to the cloud or a data center. This can help to reduce the risk of data breaches.
3) **Reduce latency:** Edge AI can help to reduce latency by processing data locally. This benefits applications requiring real-time responses, such as gaming or virtual reality, autonomous vehicles, and other mobile industrial applications.
4) **Increase scalability:** Edge AI can be scaled more easily than cloud-based AI. This is because edge AI can be distributed across many devices rather than being concentrated in a few data centers.

After conducting a comprehensive literature review, it was determined that the use of edge-AI for structural health monitoring has not yet been explored. This presents an exciting opportunity for further research in this area and could potentially lead to significant advancements in the field.

### 1) SOFTWARE FRAMEWORKS

Tensorflow, Tensorflow Lite, Caffe, and PyTorch have supported frameworks for heavy-weight and light-weight DNNs [8], [9], [23], [24]. Apache MX Net is an open-source framework for edge-based object detection and visual recognition. MLKit by Google and CoreML by Apple are mobile system frameworks used for image, text recognition, and bar-code scanning [26], [27]. In edge-AI development, it is required for the models to get converted to certain file extensions like binary files. These optimized files are compiled on hardware to get deployed. This paper utilized Tensorflow Lite and ONNX to develop an edge-compatible deep neural network model for crack detection tasks.

#### 2) HARDWARE FOR EDGE-AI

In the industry, many companies like Google, Amazon, and Microsoft are providing the edge AI service platform by enabling end devices to run ML inferences with pre-trained models locally. Various dedicated edge AI chips are commercially available like Google Edge TPU, Intel Nervana NNP, Kneron edge-AI chip, and Huawei Ascend 910 and Ascend 310 [10]. In this paper, two edge-AI platforms, i.e., Kneron KL 520 and Google Coral Dev Board, are used to perform crack detection tasks.

## III. METHODS

### A. DATASET

The public crack dataset from Çağlar and Özgenel includes concrete building crack images from METU campus buildings [20]. The dataset consists of two class labels, i.e., "Positive" and "Negative". Each class label contains 20000 images with a total of 40,000 images of cracks and non-cracks. The image characteristics of the images describe that images contain three color channels (RGB) with 227 × 227 dimensions. Fig. 2 shows the sample positive cracks from the dataset.

(a) (b) (c)

**FIGURE 2.** Sample images from the positive class of the dataset from Çağlar and Özgenel [20]. (a) shows a single crack, while (b) and (c) depict multiple cracks.

### B. WEAKLY SUPERVISED CRACK SEGMENTATION FRAMEWORK USING CRANET

CrANET is a framework for generating crack segmentation on crack images utilizing weakly supervised learning [68]. This framework includes neural network architecture built using the convolutional (Conv) and convolutional block attention module (CBAM) that performs binary image classification. This neural network architecture is trained on image-level labels and further produces crack segmentation without being explicitly trained on pixel-level labels utilizing a weakly supervised approach. The framework incorporated a two-stage process to generate the crack segmentation. Firstly, it performs the binary image classification for crack and no-crack objects. Secondly, if the image is identified as having cracks, i.e., true positive, it employs gradient class activation maps to generate heatmaps that can accurately segment the cracks.

The architecture shown in Fig. 3 has six convolution layers and three CBAM attention modules. Mishra et al. [68] performed an ablation study to devise this architecture and its hyperparameters. The first CBAM module is located between the second and third convolution layers, while the second is between the fourth and fifth convolution layers. The third and last CBAM module is placed between the fifth and sixth convolution layers. The CBAM integration adds the attention mechanism while classifying the images. After each convolution layer, there is a corresponding max pooling layer with a 2 × 2 kernel. The convolution layer, followed by the pooling layer, is then followed by flatten and fully connected layers. The input image is processed by a final layer Sigmoid activation function, which produces binary outputs. Stochastic gradient descent(SGD) is the optimizer for finding the best fit between predicted parameters and ground truths. Binary cross entropy is utilized as the loss function that compares each predicted probability to actual class output in contrast to 0 or 1. These parameters are integrated into the architecture for the training process of CrANET architecture. The batch size is 32 for the training process. The training process is designed to complete 20 epochs. This classification model is trained on the concrete cracks dataset provided by Çağlar and Özgenel [20], discussed in section III-A. The raw image size in the dataset is 227 × 227, which is pre-processed to an input size of 224 × 224 for the training process. The gradient activation maps gradCAM and guided-gradCAM are extracted from the feature maps obtained from the proposed architecture's last convolution layer (sixth convolution layer) to generate refined segmentation of cracks. The performance of the CrANET architecture in an ablation study is shown in table 1.

**TABLE 1.** Performance evaluation comparison of multiple models by Mishra et al. [68]; Train(Acc) represents the accuracy evaluation on 95% of random samples and Test(Acc) is the accuracy belongs to 5% random samples from the main dataset of [20]; *Param* represents the total number of parameters in the model; *IOUmean* describes the mean IOU score of 135 random samples labeled manually from testing sample.

| Model | Train(Acc) | Test(Acc) | Param | IOUmean |
|---|---|---|---|---|
| Simple CNN | 99.4% | 99.2% | 549K | 0.21 |
| VGG16 | 99.76% | 99.69% | 134M | 0.24 |
| ResNet50 | 99.54% | 99.42% | 23M | 0.16 |
| CrANET(soft) | 99.46% | 99.26% | 104M | 0.79 |
| CrANET(sig) | 99.6% | 99.4% | 104M | 0.83 |

Li et al. [3] discussed the lack of an open-access concrete crack dataset, especially with ground truths with segmented masks. Mishra et al. [68] also acknowledges the lack of benchmark datasets on concrete crack structures. Literature findings show that the majority of papers in crack detection are performing self-data collection, labeling, and ground truth generation processes. For example, Joshi et al. [14] have collected 3000 images of cracks and manually annotated the cracks to feed in Mask-RCNN architecture to segment the cracks. Bhowmik et al. [67] integrated robotics and deep learning by using U-NET architecture on UAVs for crack detection. They used manually annotated cracks through pixel-labeling to perform crack segmentation. A human validation study is conducted as an additional evaluation strategy where professional experts like bridge inspectors, supervisors, and civil engineers evaluate the segmentation results based on their visual perception. More details

about this can be found in section III-E. Woo et al. [17] evaluated computer vision-based visualization with human (experts) evaluation. Also, binary masks of gradCAM outputs are generated. The ground truths and binary masks from gradCAM images are used to calculate the mean IOU score. Fig. 4 shows images generated from the CrANET framework.

The CrANET framework, developed by Mishra et al. [68], is effective and reliable by utilizing a weakly supervised learning approach. The framework is less training extensive as image-level labels used to guide the learning process are responsible for learning a good representation of the data. This learning process enables CrANET to generate refined segmentation without the need for additional approaches like image thresholding techniques, which is a significant advantage. The quantitative results, including table 1, demonstrate that this approach generates refined crack segmentation. Furthermore, the CBAM attention module proposed by Woo et al. [17] is an efficient and lightweight model that is ideal for real-time applications. This feature makes it extremely useful for edge-AI developments, where dedicated ML chips have limited memory.

### C. EDGE-AI DEVELOPMENT FOR STRUCTURAL HEALTH MONITORING TASK

The section aims to build a model capable of executing inference and related computations on the edge device itself. In this section, we discuss the setup of hardware and software interfaces and the process for compatible model development for edge-AI devices. This article proposed an edge-AI framework in structural health monitoring as illustrated in Fig. 5. The framework is followed to create a highly optimized and compatible binary classifier for the edge device, which can effectively detect cracks. The edge-AI model development task replicated the binary classifier from the CrANET framework [68] using CBAM attention networks as discussed in section III-B. However, the classifier models further undergo for transformation and optimization process following Fig. 8. Two edge-AI hardware are used for platform testing, i.e., Kneron KL520 and Google Coral dev-board. These devices have compatibility constraints related to specific layers of neural networks that limit their support for certain layers. Later sections discuss the development in detail. The edge-AI development stack includes Python as a mainframe language along with machine learning frameworks like Tensorflow, Tensorflow lite, Keras, and ONNX for model optimization for device compatibility.

#### 1) KNERON KL520

Kneron KL520 chip series I used to test the edge-AI-based optimized model. This chip is assembled on a single-on-chip board. More details about Kneron KL520 can be found at URL: https://www.kneron.com/page/soc/. The documentation describes the device supports popular CNN models like VGG16, VGG19, MobileNet, etc. However, it has limited support for operators to design a CNN architecture, for example, a softmax function in the given scenario. This restricts model architecture design specific to this very device. A snapshot of the Kneron KL520 is shown in Fig. 6. From [40], brief specifications of the chip are as follows-

1) Single-on-chip (SoC) design platform
2) 32MB SDRAM (16-bit LPDDR2-1066)
3) Neural Processor Unit(NPU) for accelerating neural network process
4) CPU: Two ARM Cortex-M4, one for system control and the other as an AI co-process for NPU
5) Maximum Frequency: 300 MHz

#### 2) GOOGLE DEV BOARD

The CrANET architecture is tested in the Google Coral dev board. This single-board computer performs fast machine learning (ML) inferencing. It includes an onboard Edge TPU coprocessor that is capable of performing 4 trillion operations (tera-operations) per second (TOPS), using 0.5 watts for each TOPS (2 TOPS per watt) [39]. More details about the dev board can be found at URL: https://coral.ai/products/dev-board/. A snapshot of the Google coral dev board is shown in Fig. 7. From [39], a few brief specifications of the chip are as follows-

1) Single-on-chip (SoC) design platform
2) 1GB RAM
3) CPU: NXP iMX 8M SoC (quad Cortex-A53, Cortex-M4F)
4) GPU: Integrated GC7000 Lite Graphics
5) ML accelerator: Google Edge TPU coprocessor 4 TOPS (int8); 2 TOPS per watt

#### 3) OPTIMIZED MODEL DEVELOPMENT FOR CHIP COMPABILITY

Fig. 8 describes the flowchart and stages that led to optimized model development. These steps are followed to transform the binary classifier from the CrANET framework, illustrated in Fig. 3, to perform inference on edge devices for the crack classification task. This flowchart for developing a binary classifier involves two main operations: integrating quantizations approach to enhance the model's performance (with fine-tuning), targeting memory, latency, and power consumption, and model transformations for compatibility on the edge deployment.

##### *a: ONNX CONVERSION*

ONNX stands for Open Neural Network Exchange [51]. It is an open-source standard to represent machine learning (ML) models. ONNX is a common file format for storing ML models that allows usage for various frameworks and compilers [51]. Thus, it supports deployment to various devices with different hardware architectures. This article uses ONNX conversion to deploy the model on KL520 only. Google Coral dev-board supports model deployment using Tensorflow-Lite.

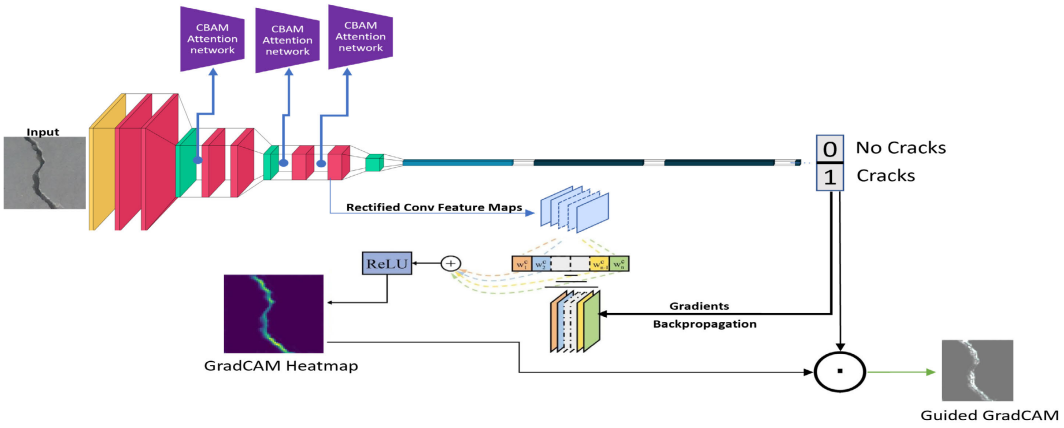



**FIGURE 3.** Weakly supervised segmentation process including binary classification, gradCAM, and guided-gradCAM approaches [68]; Red blocks are convolution layers; Green blocks represent pooling layers; Blue blocks represent flatten layers; Rest layers fully connected layers and output function.

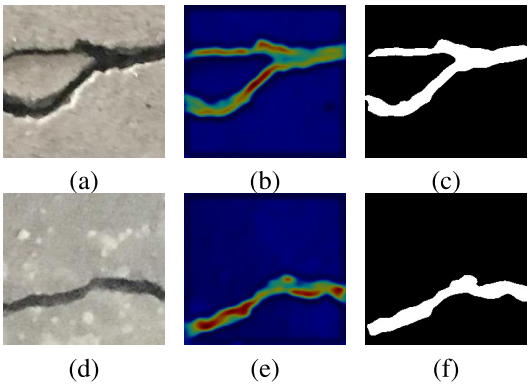



**FIGURE 4.** Crack examples with binary mask and gradCAM generated images from CrANET (sigmoid) framework [68]; *Row 1*- a) raw image b) gradCAM output c) binary mask, IOU score - 0.92; *Row 2*- d) raw image e) gradCAM output f) binary mask, IOU score- 0.87.

### *b: QUANTIZATION*

The edge devices have limited memory usage, storage size, and computing power. For an optimized model development, the quantization technique plays a crucial role in producing an optimized model. This technique involves altering the neural network to a more manageable size by implementing low-bit width numbers. Its primary objective is to decrease the size and complexity of models, thereby increasing their speed and efficiency [55]. This process reduces latency, the memory requirement, and the computational cost of neural network usage [44], [55]. In this study, quantization-aware training and post-training quantization are used to optimize the model for edge-AI devices. Post-training quantization technique quantizes the baseline model, i.e., optimizes it after being trained [54]. In this study, the model development incorporates 8-bit model transformation using post-training quantization. The quantization-aware training approach enables quantization steps during the model's training loop [54]. In this study, the model development incorporates 8-bit model transformation using quantization-aware training. These approaches allow models to reduce the model size, latency, and computation power. However, post-training quantization may enhance the efficiency of inference time, but it could also potentially lead to a decline in accuracy. Thus, it is recommended to choose between approaches after the performance comparison. This study uses a quantization-aware training approach. The details of the performances are discussed in section III-C4 in detail.

In order to produce a highly efficient model that can be used on the aforementioned edge devices, the development process entails three crucial steps that are utilized to build the edge-compatible model:

1) Perform quantization aware training: necessary to ensure 8-bit/float16 quantization required for the Edge Device (TPU/ NPU).
2) Or Perform post-training quantization: to ensure full 8-bit integer quantization required for the Edge Device (TPU/ NPU).
3) Choose the best methods from the above two based on performance considering accuracy or time inference
4) Test the model's performance using docker or simulation to evaluate results before deploying on the device.
5) Convert the original model to a device-supported file format and ensure it supports all operators used. Train the model until the desired accuracy is achieved.

Three models are built by following the flowchart illustrated in Fig. 8. All these three models are inspired by CrANET framework discussed in section III-B. The first model consists of a simple convolution neural network (CNN) with three convolution layers, while the second model is a CNN with six convolution layers. These two models are labeled as CNN (3) and CNN (6) in table 2. The binary classifier in the crANET framework with the CBAM attention network is replicated and processed utilizing the flowchart, resulting in the third model and labeled as CNN+CBAM. All the parameters, including pooling, kernel, input layer shape (227 × 227), etc., are replicated from the CrANET framework. Fine-tuned Efficient-Net is also tested to compare

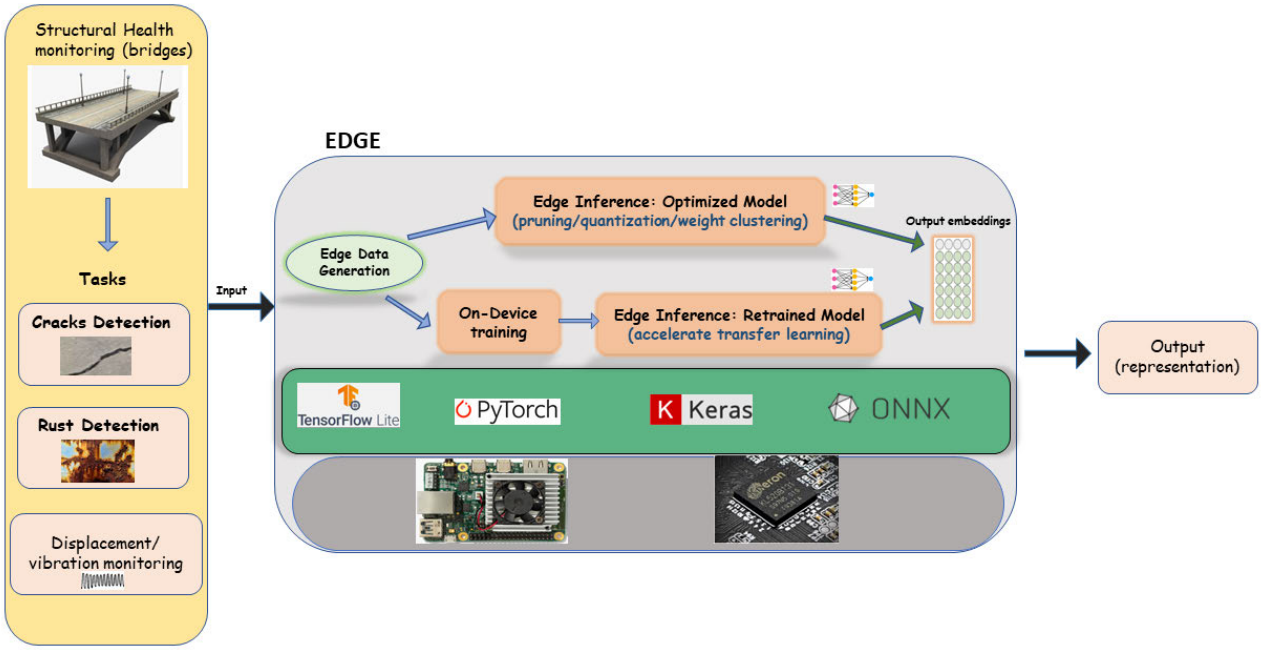



**FIGURE 5.** a) Proposed edge-AI framework in the structural health monitoring domain for tasks like crack detection, rust detection, displacement/vibration monitoring; b) The framework demonstrates the transformation into an optimized model via pruning, weight clustering, and quantization approaches; c) On-device training to support transfer learning makes it possible to accelerate inference; d) TensorFlow Lite, PyTorch, Keras, and ONNX are potential frameworks for developing optimized models on edge platforms; e) Edge-supportive data representations are operationalized to gain a desired output.

**FIGURE 6.** Kneron KL520 hardware assembly.

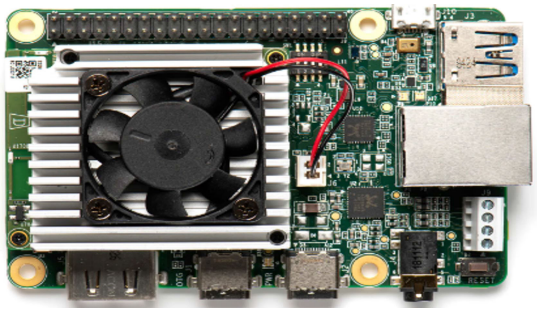


**FIGURE 7.** Google Coral Dev-board from [39].

the performance both in Kneron KL520 and Google Coral. The next section elaborates on the model performance achieved as determined by experiments.

#### 4) EDGE-AI MODEL PERFORMANCE

Table 2 and 4 illustrate the edge-AI model performance determined by the experiments. The performance evaluation of the models is done using classification accuracy. Mean inference time is also calculated to check the latency of the models. The table illustrated that CNN+CBAM, the binary classifier from the CrANET framework, achieved the highest accuracy in training and testing. The model training is designed based on training and testing sets, with a split of 36000-4000 based on 40000 images, similar to the CrANET framework's binary classifier. The training set consists of 36000 random samples, while the testing set comprises 4000 random samples from the public dataset. Interestingly, the edge-AI experiments revealed that Kneron KL520 exhibited a different mean inference time and accuracy than Google Coral Dev-board. It is worth noting that the difference in hardware specifications between these two devices could be a potential reason for the observed differences. The results in tables 2 and 4 indicate that the Google Coral dev-board outperforms the Kneron KL520 regarding classification accuracy and mean inference time. These 4000 image samples are tested using offline mode by sending image batches directly to the dev boards. The later section describes operational scenarios for each individual device.

Section III-C3 discusses quantization-aware training and post-training quantization approaches to optimize the model. It is discussed that the final model can be selected based

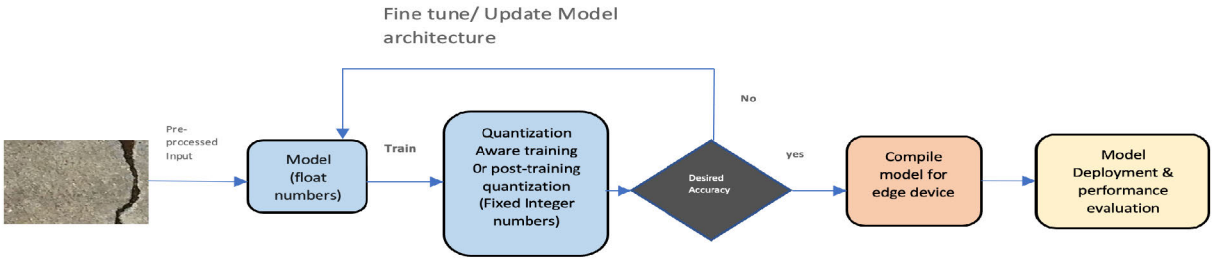



**FIGURE 8.** Flowchart outlining the various stages involved in the binary crack classification model development process on the edge devices; Quantization-aware training or post-training quantization can be opted based on their performance on accuracy and inference time.

**TABLE 2.** The time inference in the table shows the mean time taken by the devices to infer the samples from the edge-selected models using quantization-aware training (QAT) approach; TPU- is inference performance on Google Coral board, NPU- inference performance on Kneron KL520, CPU- inference performance on mainframe CPU; NA- Kneron KL520 is having limitation for the layers in CNN+CBAM model from CrANET framework thus untested.

| Model | TPU(ms) | NPU(ms) | CPU(ms) |
|---|---|---|---|
| CNN (3) | 24 | 20 | 6.7 |
| CNN(6) | 28 | 34 | 48 |
| CNN+CBAM | 350 | NA | 121 |

**TABLE 3.** Table shows the mean inference time taken by Google Coral dev board to infer the testing samples using post-training quantization (PTQ) and quantization aware training(QAT).

| Model | PTQ(ms) | QAT(ms) |
|---|---|---|
| CNN (3) | 1.04 | 24 |
| CNN(6) | 28.2 | 28 |
| CNN+CBAM | 200 | 350 |

**TABLE 4.** Table shows the training and testing performance of three different models on Kneron KL520 and Google Coral dev-board based on quantization-aware training approach; TPU(TrA)-training accuracy on Google Coral; TPU(TeA)-testing accuracy on Google Coral; NPU(TrA)-training accuracy on Kneron KL520; NPU(TeA)-testing accuracy on Kneron KL520; NA- layers in CNN+CBAM model are limited to Kneron KL520 thus not tested.

| Model | TPU(TrA) | NPU(TrA) | TPU(TeA) | NPU(TeA) |
|---|---|---|---|---|
| CNN (3) | 99.58% | 99.48% | 98.6% | 90.2% |
| CNN(6) | 99.64% | 99.54% | 99.1% | 92.3% |
| CNN+CBAM | 99.72 | NA | 99.4% | NA |

on the trade-off of accuracy and inference time. In the experiments, it was observed that post-training quantization resulted in lower average inference times compared to quantization-aware training. Table 3 shows the average inference time comparison of quantization-aware training and post-quantization training based on testing performed on the Google Coral dev board. However, the testing accuracy decreased significantly and resulted in almost 75% across all three models. The CNN+CBAM model has a precision of 65.9%, a recall of almost 99.8%, and F1 score of 79.4%. The model performs well for the crack class and significantly low scores for the non-crack class. Thus, the quantization-aware training is selected for inference for further operations. Later sections discuss the device-wise performance using quantization-aware training.

### 5) KNERON KL520 INFERENCE

The KL520 is an NPU chip on Kneron assembly shown in Fig. 6. A camera device and a mainframe system via USB are attached to the Kneron assembly for real-time testing. The assembly captures an image and sends the image into the chip to perform inference. The Kneron KL520 requires a unique process for obtaining the final inference in order to perform binary classification on image inputs. In preparation for inference, the image undergoes pre-processing on the mainframe system. This includes transforming its dimensions to 227 × 227, ensuring compatibility with the model. The models in table 4 accept an input layer of size 227 × 227, as detailed in section III-C3. Kneron KL520 requires additional compilation conversion of the onnx model into NEF format to operationalize the model prediction. More about NEF format and process to perform NEF conversion can be found on the URL: https://doc.kneron.com/docs/. Since Kneron KL520 does not support softmax and sigmoid functions, they are removed before NEF format conversion. A post-processing step is required to generate the final inference by the KL520 NPU chip. This post-processing is done on the mainframe system. This process is capable of inferencing live-stream images in real-time settings and a pool of offline images. The testing of KL520 was done for both settings. A total of 4000 random samples are tested from the public dataset, which models have not seen while training. However, the CNN+CBAM model is not tested on the KL520 due to its limitation on specific layers. The current evaluation is on the performance of a CNN model that comprises six and three convolutional layers, respectively. The study results indicate that CNN(6) surpasses CNN(3) in terms of both training and testing accuracy. The classification results of the CNN model show a training accuracy of 99.54% and a testing accuracy of 92.3%. However, table 2 reveals that CNN(3) is more efficient than CNN(6) when it comes to the average inference time per image on the device. CNN(3) takes an average of 20 milliseconds to perform inference on an image, including pre-processing and post-processing, whereas CNN(6) takes 34 milliseconds. The Kneron KL520 chip provides an efficient AI computing performance of 0.35 TOPS per watt.

#### 6) CORAL DEV BOARD INFERENCE

Google Coral dev-board is a single-board computer with fast ML inferencing, including bluetooth and wireless connectivity capabilities. It supports runs Mendel, a Debian-based Linux operating system. The device operates differently from the Kneron assembly without needing a mainframe thus, all of the necessary operations can be performed on-device. The camera can be attached directly to the board for real-time image capture up to 30 FPS. The input image is pre-processed, and the result is post-processed on the onboard CPU, while the inference is carried out on the high-performing TPU.

The testing setup and samples are the same as illustrated in section III-C5 for Kneron KL520. Table 4 reveals that the CNN+CBAM model surpasses CNN(6) and CNN(3) in terms of both training and testing accuracy. The classification results of the CNN+CBAM model show a training accuracy of 99.72% and a testing accuracy of 92.4%. Table 2 reveals that CNN(3) is the most efficient model, followed by CNN(6) and CNN+CBAM when it comes to the average inference time per image on the device. It takes an average of 24 milliseconds to perform inference on an image. Whereas CNN(6) takes an average of 28 milliseconds to perform inference on an image, including pre-processing and post-processing, whereas CNN+CBAM takes 350 milliseconds. It was observed in post-training quantization that the CNN+CBAM layers in the inference process utilize both TPU and CPU sequentially. Fig. 9 shows the CNN+CBAM model's layer utilization on the edge-TPU and edge-CPU of the Google Coral dev board. This may be a potential reason for resulting in a high average inference time both in post-training quantization and quantization-aware training. Table 2 also suggests that the Google Coral dashboard has a better average inference time for performing inference than Kneron KL520.

```
Edge TPU Compiler version 16.0.384591198
Input: cbam.tflite
Output: cbam_edgetpu.tflite

Operator                       Count      Status

SOFTMAX                        1          Mapped to Edge TPU
CONV_2D                        9          Mapped to Edge TPU
LOGISTIC                       6          Mapped to Edge TPU
RESHAPE                        1          Operation is otherwise supported, but not mapped due to some unspecified limitatior
CONCATENATION                  3          Mapped to Edge TPU
FULLY_CONNECTED                15         Mapped to Edge TPU
MAX_POOL_2D                    3          Mapped to Edge TPU
EXPAND_DIMS                    6          Mapped to Edge TPU
QUANTIZE                       2          Mapped to Edge TPU
ADD                            3          Mapped to Edge TPU
MEAN                           5          Mapped to Edge TPU
MEAN                           1          More than one subgraph is not supported
MUL                            5          Mapped to Edge TPU
MUL                            1          More than one subgraph is not supported
REDUCE_MAX                     1          More than one subgraph is not supported
REDUCE_MAX                     5          Mapped to Edge TPU
```

**FIGURE 9. a) Google Coral Dev-board inference for post-training quantization utilizing both TPU and CPU and resulting in higher inference time; b) If a layer is not mapped to edge-TPU then by default it is mapped to edge CPU.**

The results obtained from the experiments conclude that edge-AI devices excel in the task of crack classification in live-streamed images and offline images with significant classification scores and average inference time. The conducted experiments also validate the effectiveness and practicality of the proposed edge-AI framework as depicted in Fig. 5. The next section III-D describes the integration of edge-AI with cloud computing to enhance the usability and implication of the proposed edge-AI framework. This integration utilizes Google Coral board along with CNN+CBAM and CNN(6) for further experiments based on their hardware design and classification accuracy strength, respectively, as shown in table 2 and 4. Fig. 10 shows the instance of crack image classification using the Google Coral dev board.

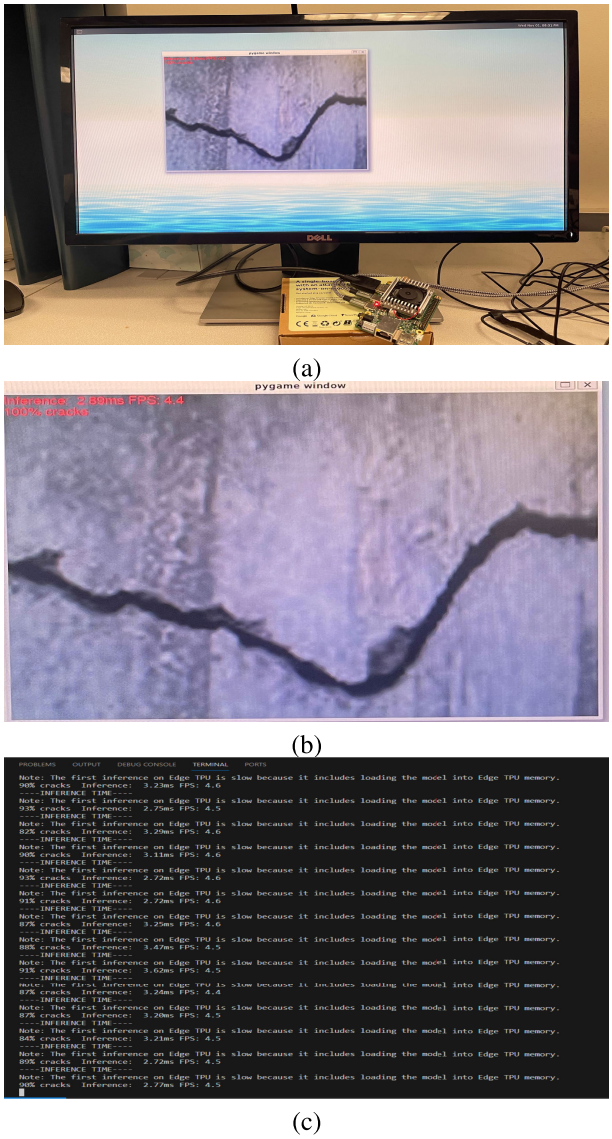



**FIGURE 10. Google Coral Dev-board inference using EfficientNet after on-device training on live-streaming of cracks on a concrete bridge; On-device training for adaptive approach is discussed in section III-F; a) The image shows the real-time classification scores for the crack image classification task; b) output view of a); c) The setup demonstrates the real-time inference on the console.**

### D. CLOUD-EDGE ADAPTIVE INTELLIGENCE FRAMEWORK

This section proposes a cloud-edge adaptive intelligence framework by integrating cloud computing capabilities along

with the proposed edge-AI framework to enhance usability and effectiveness on crack detection and quantification in real-time physical SHM sites. Section I discusses the thoughts of Zhou et al. [32] and Mendez et al. [31] claiming the best level of edge intelligence is application-dependent. The bottleneck in weakly supervised segmentation performance due to minimal memory and limited operator support led to the proposal of this framework integrating edge-AI and cloud resources. Fig. 13 illustrates the proposed framework in the SHM domain. This framework comprises two crucial stages: image classification inference on the edge and segmentation and quantification on the cloud. These stages sequentially generate final outputs, including crack classification score, segmentation, and quantification of cracks for a given input image. The framework proposed here is also designed to facilitate the integration of crowd intelligence by utilizing human-in-the-loop feedback and adaptive AI. By combining these characteristics, the framework aims to create a more effective and efficient system that can adapt to changing circumstances and achieve better results. The experiments done in this section are conducted using Google Coral dev-board and Amazon AWS cloud services.

A responsive web application is developed to operationalize and execute the proposed CEAIC framework. The application takes an image input and performs tasks, including classification, segmentation, and quantification, sequentially. Once these outputs are generated, the user can also record their feedback following human-in-the-loop characteristics. Further, the generated outputs and the human feedback are processed for adaptive AI setup. Fig. 11 demonstrates the snapshot of the developed web application. The later section describes how these characteristics and stages are integrated, operationalized, and utilized on a physical concrete bridge site following the proposed CEAIC framework. There are limitations in the use-case of this web application. These limitations are discussed in detail in section IV-C.

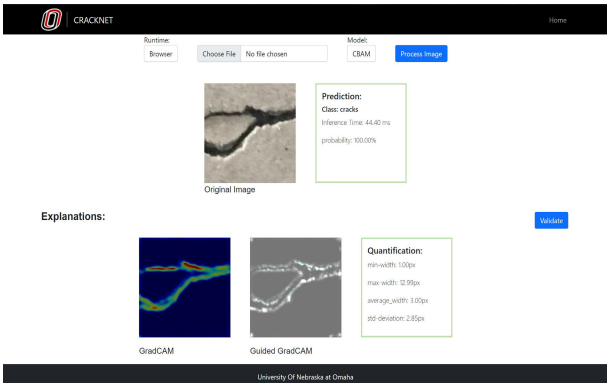



**FIGURE 11.** Snapshot of the responsive web application developed following the proposed CEAIC framework; The application is capable of taking a live-stream (online) image using a camera and also accepts offline images; The bottom images are generated by CrANET framework explaining the segmentation of the cracks in the image along with quantification measures;The validate button is the function where user can record their feedback.

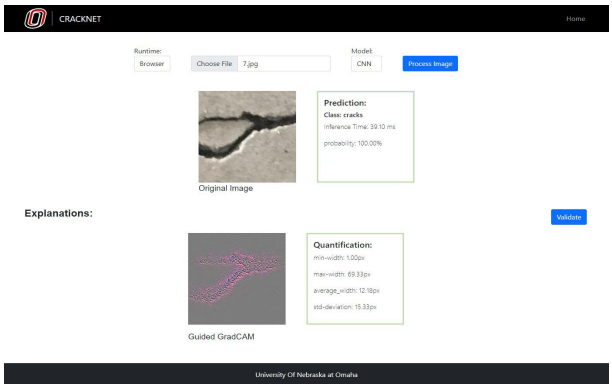



**FIGURE 12.** Segmentation generated by gradCAM approach using CNN(6) using cloud resources; The segmentation generated by the CNN(6) are noisy; Addition of CBAM enhances the segmentation as seen in Fig. 11.

### 1) IMAGE CLASSIFICATION ON THE EDGE

This article outlines various options for determining an optimal approach for deploying the models discussed above, considering the limitations of edge devices used in this study. These devices have minimal memory and supported operations, which can create performance bottlenecks while processing resource-intensive tasks like segmentation or engaging multi-user scenarios. A series of experiments were conducted to generate segmentation using the weakly supervised approach on edge devices. It is found that the gradCAM approach for weakly supervised learning is not a well-suited process on edge devices, but as per the dev-board document, supervised learning segmentation is possible on edge devices. The article targets to perform the image classification task at the edge-AI platform to implement an efficient system considering the bottlenecks. The possible deployment options include deploying the model on an edge device, sending the image from the host to the device for inference, or utilizing an edge device like a mobile web browser.

This article acknowledges the potential Google Coral dev-board, to deploy models from the cloud seamlessly. Cloud-edge integration provides a seamless and efficient way to transmit data from an edge device to the cloud, making it easier to manage and analyze data remotely. The web application developed allows one to click/feed the image on the bridge site. After capturing the image, it is sent to the edge-AI platform to execute the classification task. The input is sent to the cloud for segmentation and quantification after being classified as cracks. The connection of the web application using a smartphone with the Coral dev board is established using Bluetooth connectivity. The experiments conclude that Bluetooth connection is not reliable. Section IV-C and IV-B discuss this bluetooth reliability issue in detail. Google Coral dev-board offers on-device training capabilities that can be utilized for Adaptive AI capability. This is outlined in the section.

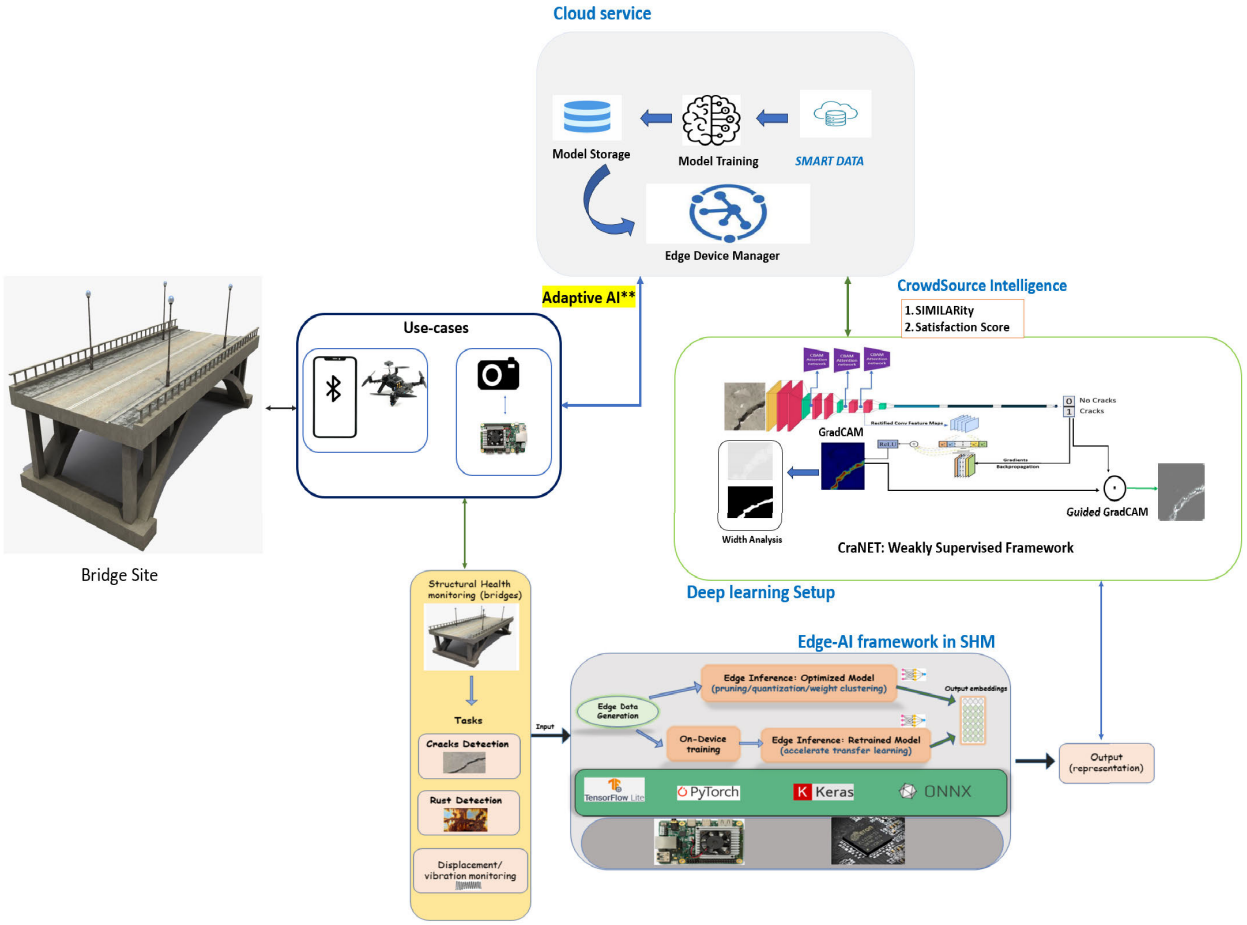



**FIGURE 13.** Proposed Cloud-edge adaptive intelligence framework (CEAIC) for cracks detection and quantification in the structural health monitoring domain using weakly supervised segmentation; In this proposed framework, there are four main components, namely deep learning setup, cloud resources, edge-AI model development, and use-cases.

### 2) SEGMENTATION ON THE CLOUD

The cloud services from Amazon AWS services are utilized to perform the tasks, including segmentation and quantification. The above section discussed the image classification task and execution once the image is uploaded. After performing the classification task, if the result reveals a true-positive class, i.e., the cracks, then further operation is offloaded to the cloud. In practice, images are sent to an API endpoint hosted on the cloud, where they are processed using CrANET framework from section III-B to create segmented outputs and quantification data. In the CEAIC framework, CNN (6) and CNN+CBAM are implemented. CNN+CBAM is used for the final segmentation due to its efficient performance, as discussed in section III-B. The CNN+CBAM in the CrANET framework generated two distinct segmentations using explainable AI approaches, which are gradCAM and guided-gradCAM. The gradCAM image undergoes further processing, leading to the generation of a binary mask for quantification purposes to compute the width analysis of the crack, as illustrated in section III-D3. Fig. 3, 4, and section III-B discuss the crack segmentation process in detail. This cloud-centric approach ensures efficient handling of resource-intensive tasks of gradCAM to segmentation in the cloud environment.

### 3) CRACK QUANTIFICATION

In the framework, as shown in Fig. 13, one of the tasks includes quantification of cracks, i.e., analysis of cracks based on width. The proposed approach for quantification involves turning the segmented results into a binary image using image processing methods. The quantification process on the binary image consists of three steps, including distance transform, skeletonization, and crack width measurement. Fig. 14 demonstrates the flowchart for the crack quantization process. It is important to clarify that the primary objective is to evaluate the efficiency of the proposed framework, rather than focusing on the implementation details of the width measurement. Although the proposed approach for quantifying cracks serves the purpose of crack quantification, there are several alternative techniques available, each with its own strengths and applications that can be explored as future work for specific use cases.

#### a: DISTANCE TRANSFORM

The first step involves the application of a distance transform to a binary image of segmented cracks. This transform calculates the distance of each pixel within the crack to its nearest background pixel [10]. The distance map generated provides accurate information on the exact distance of each point within the crack from the closest edge.

#### b: SKELETONIZATION

Subsequently, skeletonization is employed to refine further understanding of the crack's structure. Skeletonization is a process that reduces the binary mask of the crack to a one-pixel-wide representation, emphasizing the core of the crack while preserving its essential characteristics [28].

#### c: CRACK WIDTH MEASUREMENT

By superimposing the skeleton on the distance transform output, It is possible to compute the thickness of the cracks at individual points. This thickness data is pivotal for quantifying the dimensions of the cracks within the segmented area. The key metrics, such as the minimum, maximum, average, and standard deviation of the crack widths, are also computed.

The proposed CEAIC with cloud-edge integration also incorporates human-in-the-loop(HITL) as an approach to crowd intelligence to enhance the model performance and evaluation strategy. Details are provided in section III-E. Further, the HITL approach is also helping to practice adaptive learning in the proposed framework of CEIAC. Section III-F provided comprehensive insights about adaptive learning integration. The use-cases CEAIC using web application is discussed in the next section IV.

### E. CROWD INTELLIGENCE: HUMAN-IN-THE-LOOP

This study incorporates two practices for improving the CrANET framework for weakly supervised segmentation, i.e., crowd intelligence and human-in-the-loop(HITL). Crowd intelligence is the ability of a group of people(crowd) to solve a problem [18]. In machine learning (ML), crowd labeling is used to enhance model performance by incorporating the crowd for data labeling, performance evaluation, and bias mitigation [23]. The integration of the HITL approach as crowd intelligence can lead to significant improvements in ML model performance [7], [23]. Additionally, this approach can help to identify and overcome potential biases and limitations, thus making the model more accurate and reliable.

As discussed above and in the article by Mishra et al. [68], the human validation study is utilized as HITL strategy (apart from IOU scores) to evaluate crack segmentation from the CrANET framework. In this study, 21 professionals participated, including bridge inspectors, supervisors, bridge designers and engineers, scour inspectors, etc. Most of these professionals have more than ten years of professional experience in civil engineering and structural health monitoring domains. Thirty-five (35) sample images are randomly selected from the predicted true positive samples from the testing sample. A total of 733 votes are counted out of 735 ($21 \times 35$) in this validation study from 21 professionals for 35 questions. The aim of the human validation study is to provide expert opinion about the degree to which the predicted gradCAM visual heatmap (shown in Fig. 4) is similar to the raw image. A construct SIMILArity is proposed to objectively measure expert's opinions as HITL approach. The construct *SIMILArity* is proposed to assess the conformance of the location, pattern, and shape of the cracks given in the raw image and predicted image. Almost 71% of votes are given for exactly similar and somewhat similar by participants in the study. Based on the findings of the human validation study and the mean IOU score presented in table 1, it can be concluded that the proposed approach is a highly effective framework for image segmentation using image-level labels. More details of the human validation study for the CrANET framework can be found in the discussion of Mishra et al. [68].

The proposed CEAIC framework in this article also incorporated a crowd intelligence approach to evaluate and validate the output generated by the framework. The framework utilizes the same human evaluation strategy that was discussed earlier to evaluate segmentation from the CrANET framework. The use cases of the crowd intelligence incorporating HITL system are discussed in detail in the next section IV.

### F. ADAPTIVE AI

The Google Coral dev-board and Kneron KL520 are used as edge-AI platforms to conduct experiments for crack detection tasks. The documentation for both devices claims they can perform on-device training using transfer learning and fine-tuning [39], [40]. This approach implements on-device training for the adaptive AI model to learn and adapt to new data. The limitations of the devices are discussed in Section III-D. These devices have certain limitations that prevent them from implementing the adaptive AI approach proposed in the CEAIC framework. The Google Coral development boards offer support for on-device training. According to the documentation, Coral board only supports pre-built models from their repository for on-device training and does not support custom models [39].

This article focuses on performing on-device training using pre-built models from the Coral dev board's repository. EfficientNet is used to conduct on-device training. For testing purposes, 400 new images are used, comprising 250 images of cracks and 150 images of non-cracks. These 400 test samples for both positive and negative classes are collected manually by the authors in real-time testing. The manually collected samples impact testing accuracy with only 75% due to different data distributions and variations in crack characteristics, including length and width, and background textures. Fig. 10, 19, and 20 are instances of

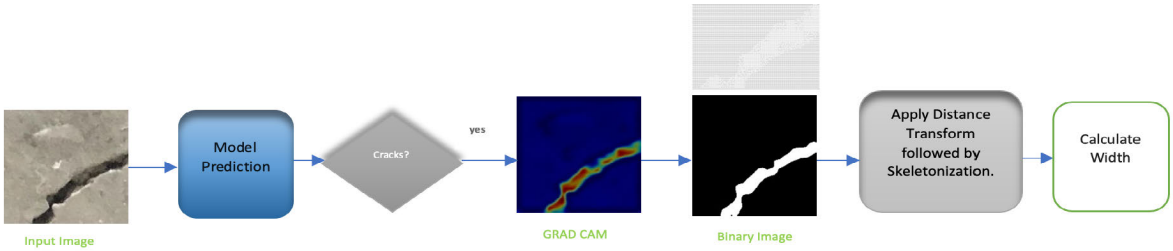



**FIGURE 14.** Flowchart outlining the various stages involved in crack quantification task utilizing the cloud resources; Once the image is segmented it undergoes binary image transformation followed by distance transform and skeletonization.

manually collected cracks. The testing accuracy is improved to 95.2% by utilizing on-device training by retraining the model using the transfer learning approach as proposed in the edge-AI framework in Fig. 5. The Efficient-Net model also achieves 2.5 milliseconds of average inference time for 400 testing samples. These results indicate that the model's performance is enhanced through on-device training using transfer learning. This outcome serves as evidence for the proposed CEAIC framework for adaptive AI.

## IV. DISCUSSIONS

### A. USER INTERACTION IN REAL-WORLD LAB ENVIRONMENT

This section outlines a user's steps to interact with real-world scenarios following the CEAIC framework discussed in section III-D. The created lab settings replicate a real-time scenario with cracks on the bridge. However, the setting follows the CEAIC framework but does not include a human feedback loop or an adaptive AI approach due to memory limitation. A USB camera is attached to the Google Coral dev board that takes online images in real time. After the user selects the perfect crack frame, the system sequentially performs crack detection tasks, including classification, segmentation, and quantification. Fig. 15 describes the lab settings performing crack detection tasks. Section III-D describes how crack detection is performed using a Google Coral dashboard for crack image classification on the edge, while weakly supervised segmentation and quantification are generated using cloud resources. The lab settings displayed in Fig. 15 imitate similar tasks on the Google Coral dev board and Amazon AWS cloud. The crack detection task is completed in an average of 13.5 seconds per image to generate results, as shown in Fig. 15, based on 50 test samples. Users can adapt this setting to test images in offline mode.

### B. USER INTERACTION ON WEB APPLICATION

This section outlines a user's steps to interact with the application to perform the crack detection task and record the feedback. Section III-D demonstrates the two-stage process in the proposed CEAIC framework - image classification on the edge device, segmentation, and quantification on the cloud service, with periodic data transmission. The steps to operate the developed web application are described below.

The web application is being operated using an iOS device. The Google Coral dev-board is connected to the iOS device for image transmission using a bluetooth connection. These settings are tested on a physical bridge site in Lincoln, Nebraska. The test experiments on the site conclude that the bluetooth connection is unreliable. The connection kept dropping frequently while transmitting the image data on the edge device from an iOS platform. Ten images are tested using a bluetooth connection on the physical bridge site. The details of reliability issue using Bluetooth is discussed in detail in section IV-C. The testing of ten image samples showed that it takes an average of 16 seconds to generate results for one image.

#### 1) IMAGE INPUT

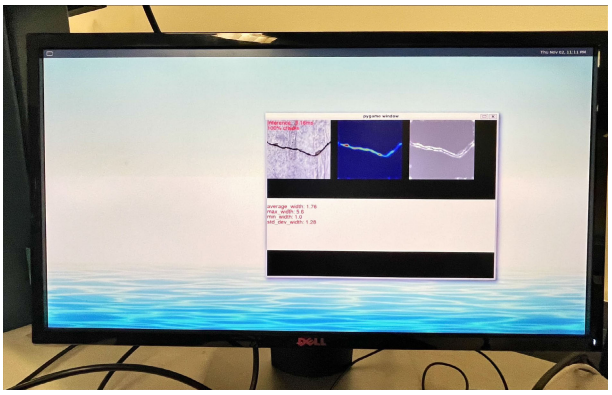
Fig. 16 demonstrates the web app in an iOS device requesting an input image. The application allows users to capture images in real time using the camera and upload images saved offline. Furthermore, the images in a digital access management system can also be fed to an app for inference. If the user is accessing the camera of the iOS/Android device, they can adjust the zoom to feed a desired input. Offline image integration also incorporates the same functionality. Users can adjust cropping before feeding them into the app. The limitation in section IV-C discusses zoom adjustment and offline cropping.

#### 2) MODEL SELECTION

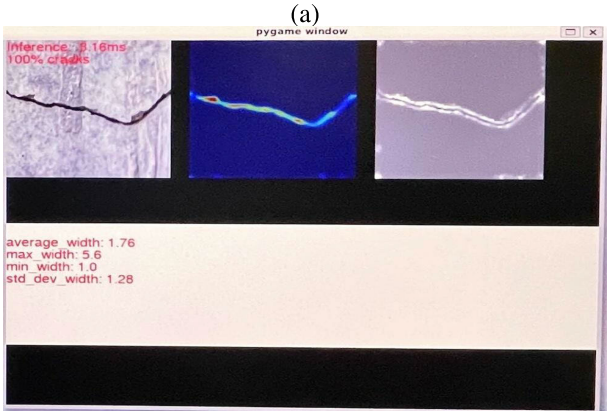
Once the image request is accomplished, the user can proceed to perform model selection. Based on their performance, the app has integrated only CNN (6) and CNN+CBAM from the CrANET framework. CNN+CBAM model is recommended for use against CNN(6) to achieve the best outcome, as table 1 in section III-B validates the classification and segmentation performance of CNN+CBAM.

#### 3) INFERENCE

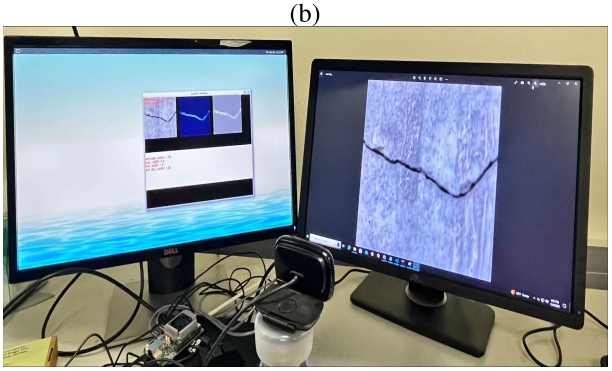
After model selection, the user must press the *Process Image* button to generate inference for the image. This process performs binary image classification for the image input on the Google Coral dev board through bluetooth image data transmission. The inference results include three metrics- class, inference time, and confidence score. The class defines whether image input belongs to the cracks or



(a)




(b)



(c)

**FIGURE 15.** Crack detection task demo in a lab environment; This setting does not include the human-feedback functionality; a) The image shows the real-time generated outputs for the crack image; b) Output view of a); b) The setup demonstrates the real-time use case with edge camera that is adjusted to take a perfect snippet of the crack image.

non-cracks category. The inference time unit is milliseconds, representing the time it takes for the Google Coral Dev-board to complete an inference. The confidence score indicates how certain the model is about the accuracy of the image classification and is represented with probability percentage. Fig. 17 shows the inference result generated by the web app once the image input is processed.

#### 4) SEGMENTATION AND QUANTIFICATION

The *Process Image* button processes the classification task, segmentation, and quantification sequentially. The *Explanation* column in the web app displays the generated results

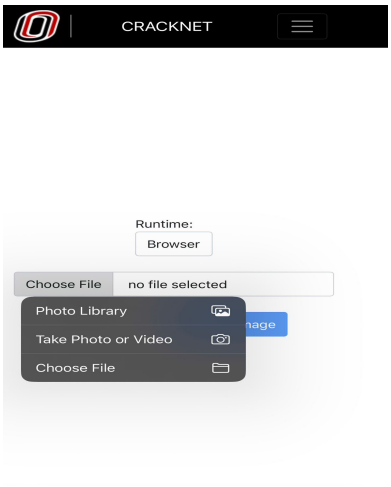



**FIGURE 16.** Web application in an iOS device requesting an input image.

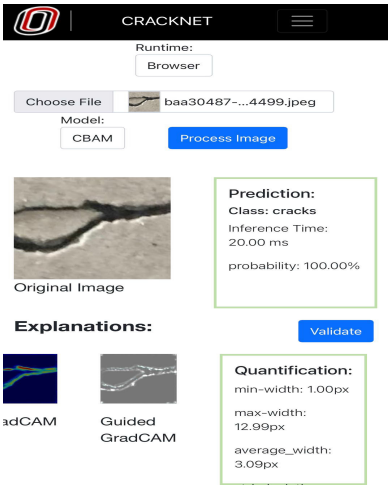



**FIGURE 17.** Potrait mode of web app in an iOS platform demonstrating results generated for segmentation and quantification; Bottom row-Left image: displays the segmentation generated by gradCAM; Middle image: displays the segmentation by guided-gradCAM; Right box: shows the quantification metrics describing the width analysis in pixel units.

after the classification task. The segmentation results in outputs using gradCAM and the guided-gradCAM approach. The segmentation and quantification task together takes an average of 12 seconds, based on 100 image sample tests (only tested for segmentation), to execute the final results. The quantification task is performed by utilizing the segmentation generated by the gradCAM approach. In the AWS cloud, the binary mask is generated using grayscale image transformation and undergoes width analysis discussed in section III-D3. The quantification results yield four metrics representing the width analysis, including minimum, maximum, and average width and standard deviation. These metrics are calculated in pixels as a unit. Fig. 17 illustrate the segmentation and quantification results.

#### 5) HUMAN VALIDATION

A crowd intelligence approach utilizing HITL feedback is practiced in the app. More details can be found in

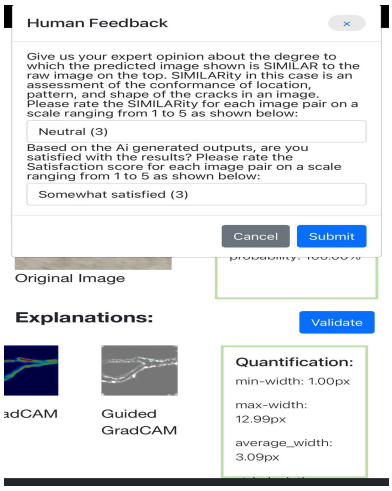



**FIGURE 18.** User's feedback on app using iOS platform; User's feedback options based on Likert Scale for SIMILARity and satisfaction score constructs;.

section III-E. Users have to press *Validate* button to record their feedback input. They can provide objective feedback based on visual perception by comparing raw and generated images using a construct, SIMILARity. This construct is replicated from the CrANET framework as discussed in section III-B and III-E. Additionally, satisfaction score, which is objective feedback, is also integrated with the HITL feedback system. Users can provide this feedback using a Likert scale that measures their perception of generated results, including images and crack quantification. Users can press the *Submit* button to record their feedback. Fig. 18 demonstrates the HITL feedback in the web application.

### C. LIMITATIONS AND FUTURE WORK

Section IV-B demonstrates the user interactions on the lab settings and web application to perform crack detection and quantification utilizing edge-AI and cloud computing capability. The crANET framework in section III-B and the quantification approach in section III-D3 are integrated into the web application to generate final outputs. This section clearly identifies the importance of acknowledging the study's limitations and providing a better understanding of the research's scope and potential implications.

#### 1) CRANET FRAMEWORK

Fig. 5 and 13 illustrate the proposed edge-AI and cloud-edge adaptive intelligence framework in the SHM domain incorporating the CrANET framework. Section III-B and Mishra et al. [68] explains the objective of the proposed CrANET framework is to perform crack segmentation using image-level labels in a weakly supervised manner. However, they observed a few failure modes in using the gradCAM approach. The observation determines that gradCAM has limitations when dealing with images that contain complex backgrounds, i.e., having a variance of color saturation on concrete surfaces based on surface finish and wide cracks in the image. Çağlar and Özgenel [20] acknowledge the variance in concrete surfaces. From exploratory data analysis, it is observed that these images have a limited presence in the distribution. As part of future work, increasing the samples of these images utilizing adaptive AI and crowd intelligence can potentially overcome this limitation.

**FIGURE 19.** Crack on concrete bridge column present under the bridge deck.

#### 2) REAL-TIME TESTING USING WEB APPLICATION

Zoom and crop adjustment functions for the live stream and offline image input are discussed in section IV-B. In this study, it is important to acknowledge this functionality considering the data distribution (data behavior) used in model training for the proposed CEAIC framework. The machine learning literature recommends that training and testing data should belong to the same distribution in machine learning [8], [12], [64]. ML models learn to predict patterns in training data. If the testing data comes from a different distribution, the model may not be able to generalize well to the new data. The images were clicked approximately 1m away from the surface. They claimed the images were captured on the same day with similar illumination conditions. Fig. 2 are the samples from the dataset proposed by Çağlar and Özgenel [20]. The observation in exploratory data analysis suggests that training samples which is 36000 samples contain similar cracks characteristics as shown in Fig. 2. In this context, crack characteristics in the image are defined by patterns of crack progression, width, and background color. Fig. 19 shows a crack on a concrete bridge column present under the bridge deck. This crack image appears to differ significantly from the dataset from Çağlar and Özgenel, used to train the model. Interestingly, the web app classified the image as cracks with a confidence score of 99.29%. However, the segmentation results are noisy and do not represent the crack localization. Fig. 21 demonstrates the results of the image shown in Fig. 19. Using the cropping (or zoom adjust for online images) function, a portion of the crack is cropped to imitate similar crack characteristics from the training dataset as shown in Fig. 20. This image produces a significant outcome when processed in the app, as demonstrated in Fig. 22. The results generated from this image input validate the accuracy and effectiveness of the app. Thus, it is recommended to feed image inputs that possess similar crack characteristics

**FIGURE 20.** Cropped crack portion belongs to crack shown in Fig. 19; The cropped image includes almost similar crack characteristics from the training sample.

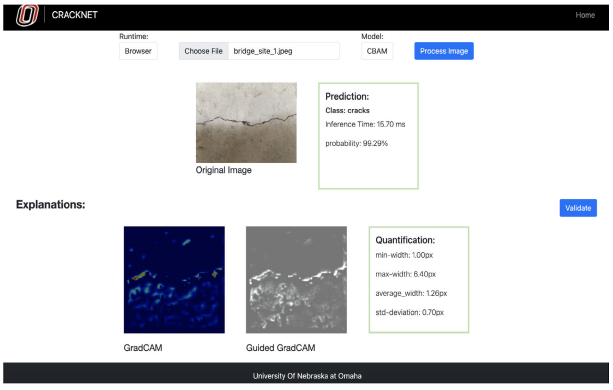



**FIGURE 21.** Results for the bridge crack from Fig. 19.

to ensure the best possible performance from approaches proposed in the current CEAIC framework.

However, the literature also acknowledges that training and test data with similar distribution is seldom the case in a real-world setting, thus, transfer learning came into practice [8]. Transfer learning is used to generalize an ML model for the new data distribution. This can be accomplished by gathering data from various sources or through data augmentation techniques that generate similar data to existing data [8], [64]. Transfer learning is a potential future work of this study to include crack data from various locations, backgrounds, colors, and textures.

#### 3) WEAKLY SUPERVISED SEGMENTATION ON EDGE DEVICES

Section III-D1 describes the binary image classification task on cracks image using edge-AI capability. The section also discusses the edge-AI device capability to operationalize the crack detection task including segmentation, especially weakly supervised segmentation. The explanation includes that edge-AI devices have minimal memory and supported operations, which can create performance bottlenecks while processing resource-intensive tasks like segmentation or engaging multi-user scenarios. Several experiments were carried out to generate segmentation using the gradCAM method on edge devices. It was discovered that the gradCAM method is not well suited for weakly supervised learning on edge devices. The segmentation results generated by

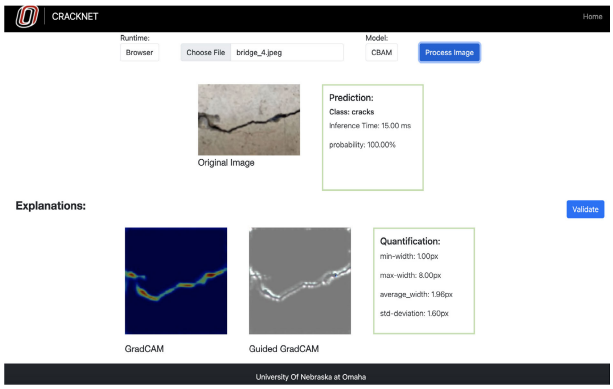



**FIGURE 22.** Results for cropped crack portion from Fig. 19.

gradCAM are noisy as shown in Fig. 12. The Google Coral document claims that dev-board can generate segmentation by supervised ML approach [39]. Potential future work motivated by this study is to explore the possibility of the limitation of generating WSL-based segmentation on-device itself. This will allow generating results all-on-device rather than using cloud services to generate weakly supervised segmentation.

#### 4) DATA TRANSMISSION ON EDGE-DEVICE

Section IV-B and III-D explains the connectivity issue using bluetooth for transmitting the image from an iOS device to Google coral dev-board. It was observed that the bluetooth connection frequently dropped during image transmission while testing on the physical bridge site. The testing was done only on ten image samples. Fig. 20 and 19 shows the instances of testing on a physical bridge scenario. A native application development utilizing the mobile GPU as an edge device is a potential future work to overcome this connection issue. This future work does not involve any need for transmitting data through the application.

### *D. IMPLICATIONS*

#### 1) ASSESSING CEAIC TRUST

This section discusses the factors that support the effectiveness and trustworthiness of the proposed CEAIC framework in section III-D. The proposed cloud-edge adaptive intelligence framework for performing crack detection and quantification integrates edge-AI framework in the SHM domain, weakly supervised crack segmentation approach, and crowd intelligence approach to practice human feedback. Mishra et al. [68] discuss characteristics of the CrANET framework to asses its trust by referencing Selvaraju et al. for the gradCAM approach. They highlighted the benefits of using gradCAM visualization, which can provide valuable visual explanations to understand and interpret data-driven decisions. The gradCAM observations offer valuable insights into how neural networks function to detect cracks in images accurately. They advocate that the gradCAM feature in the CrANET framework can effectively identify the regions of interest within an image that are being targeted by the neural

network during binary image classification tasks. It has been suggested by Selvaraju et al. [38] that the gradCAM approach can be a valuable tool for users to enhance their evaluation of automated systems and to foster end-user confidence. Table 1 in section III-B illustrates the performance evaluation of models and segmentation evaluation using meanIOU scores and human validation study. This concludes that the implementation of weakly supervised learning techniques would greatly benefit the task of crack segmentation. Thus, the proposed two-stage CEAIC framework, including edge-AI for binary image classification tasks and cloud service to perform segmentation, quantification, and adaptive AI based on crowd intelligence, can help users and domain experts place trust

### 2) IMPLICATIONS FOR PRACTICE

Fig. 13 illustrates the potential use-case of the framework implementation in the SHM domain. The framework shows use cases of the framework incorporating drones, mobile/tablets, and smart cameras. The article provided comprehensive insights into the framework by showcasing two practical and effective use cases via a responsive web application. Firstly, using the web application on the mainframe system as shown in Fig. 11 and 12, secondly on the iOS platform using a mobile system that has a camera function for live-stream (online) images and also utilizes digital access management system for offline images.

The economic viability of implementing the proposed deep learning models on the edge is primarily related to the hardware or assembly on which these models would be run for monitoring large bridges such as UAVs, ML chips, and mobile devices. These supplementary costs must also be considered in addition to training costs for new ML models. The impact of AI chip developments on hardware costs depends on various factors, such as the type and scale of AI workloads, the availability and demand of AI chips, and the innovation and differentiation of AI chip makers [69]. However, this study anticipates that hardware costs may rapidly decrease with new AI chip developments.

A potential use case for the CEAIC framework is its integration with drones. This integration proves to be a valuable solution for bridge locations that are out of reach for humans, allowing bridge health inspectors to address and resolve any issues in those areas effectively. This integration will potentially help to mitigate any risks to human life. The integration can be accomplished by mounting the edge-AI device on the drone. The device's compact size, as shown in Fig. 6 and 7, which is almost the same as a credit card, makes it easy to carry it around. Edge-AI device mounting can significantly improve the maintenance of bridge decks and other inaccessible areas, ensuring their longevity and safety.

In this article, the authors introduce the edge-AI and CEAIC framework in the Structural Health Monitoring (SHM) domain. The main purpose of this framework is to perform crack detection and quantification tasks. In addition, Fig. 5 shows that the proposed edge-AI framework can also be used for other tasks such as rust detection, displacement monitoring, and vibration monitoring on a bridge. The CEAIC framework usage within other areas of the SHM domain can be another crucial implication.

Section IV-C discusses the limitations of edge-AI devices practiced in this study to perform crack detection tasks. The edge-AI devices have limitations in performing weakly supervised segmentation and performing data transmission. A potential future work that can rectify this limitation will enable the edge-AI device to have full capability to perform crack detection tasks completely, including on-device training. This implication will not only reduce the usage of cloud computing resources but will enhance the data offloading, privacy, and latency in result generation. The development of native applications for a smart (mobile) device that utilizes mobile GPUs and resources can be a potential implication of practice to perform all-on-device crack detection task.

### 3) IMPLICATIONS FOR RESEARCH

This article utilized a weakly supervised segmentation approach to perform a crack detection task considering image-level (weak) labels rather than pixel-annotated (strong) labeling. The literature claims that most of the data in the world are unlabeled [26], [47], [48]. One of the reasons is that labeling data can be a challenging and time-consuming task. Mishra et al. [68] and Hamishebahar et al. [2] highlight the importance of label-free and weak-label algorithms in Structural Health Monitoring (SHM). This is due to the large amount of unlabeled data in SHM and the need to minimize human intervention, as human annotations and labeling can be expensive and introduce bias. Li et al. [3] acknowledges the lack of an open-access concrete crack dataset, especially with ground truths with segmented masks. The literature for crack detection tasks in the SHM domain suggests that most papers in crack detection are performing self-data collection, labeling, and ground truth generation processes [68]. In the context of structural health monitoring (SHM), the development of label-free algorithms holds significant implications.

Unsupervised learning algorithms are designed to identify patterns and relationships in unlabeled data without the need for human guidance. This type of machine learning is particularly useful for discovering hidden insights and making predictions based on raw data. In recent years, self-supervised learning has become increasingly popular due to the abundance of unlabeled data available. Self-supervised learning(SSL) enables models to learn from unlabeled data through tasks that do not require human labeling [58]. The SSL implementation includes contrastive learning algorithms, which can distinguish between cracked and non-cracked images by learning their differences. Liu et al. [57] explain that SSL algorithms perform two stages of training. Initially, the algorithm is trained on a huge

set of unlabeled data using a pretext task or a contrastive learning algorithm. After the algorithm has been trained, it can perform various downstream tasks, such as image classification or detecting cracks in tasks. Vision transformers (ViTs) are a neural network type utilized for self-supervised learning that achieves state-of-the-art results on various computer vision tasks, including image classification, object detection, and semantic segmentation [49]. ViTs can extract and learn features that represent the visual features of images. These features are utilized in visual representations that aid in various computer vision tasks, such as classification and segmentation tasks. DINOv2 and DINO are self-supervised learning methods for training vision transformers (ViTs) to perform robust, efficient, and state-of-the-art on a variety of computer vision benchmarks, including image classification, object detection, and semantic segmentation [29], [35]. These have significant implications for performing crack detection tasks in the SHM domain given the context of the large amount of unlabeled data in SHM and the need to minimize human intervention, as human annotations and labeling can be expensive and introduce bias.

## V. CONCLUSION

The aim of this study is to uncover the untapped potential of edge-AI approaches in the domain of structural health monitoring (SHM). The article explored the edge-AI approach by conducting experiments using edge-AI devices on crack detection tasks and proposed two frameworks. Firstly, an edge-AI framework that demonstrated the integration of edge-AI to perform inference on SHM tasks. Secondly, a cloud-edge adaptive intelligence framework to perform crack detection (CEAIC) in real-time physical sites. This CEAIC framework incorporates the proposed edge-AI framework, the crowd intelligence approach, and the CrANET framework to perform weakly supervised crack segmentation. The main objective of weakly supervised crack segmentation is to segment cracks based on image-level labels with minimum human interventions to reduce human cost and labeling time and prevent bias due to incorrect human annotations and labeling.

Kneron KL520 and Google Coral dev-board are used as platforms to operationalize edge-AI capabilities. The edge-AI platform only performs binary image classification for crack detection due to minimal memory and supported operations, which created performance bottlenecks while processing resource-intensive tasks like segmentation using weakly supervised learning and engaging multi-user scenarios. Thus, Amazon AWS cloud services are utilized to integrate cloud capabilities to perform weakly supervised segmentation, quantification, and recording human feedback tasks. The average inference time to perform binary image classification from the CrANET framework is 49 milliseconds taken by the Google Coral dev board. The segmentation and quantification task together takes an average of 12 seconds (based on 100 image tests) to execute the final results. A responsive web application is implemented following the proposed CEAIC framework that successfully tests its effectiveness and provides validity. The user interaction is demonstrated by the web application to execute crack detection tasks on real-bridge sites. This research will motivate researchers and practitioners to integrate edge-AI approaches in the SHM domain to enhance productivity in real-time scenarios.

## ACKNOWLEDGMENT

This article acknowledges Mishra et al. [41], Mishra et al. [68], and poster and presentation of CEAIC framework in Bridging Big Data Workshop 2023 conducted at the University of Nebraska, Lincoln [53].

## REFERENCES

[1] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "Inference-aware convolutional neural network pruning," *Future Gener. Comput. Syst.*, vol. 135, pp. 44–56, Oct. 2022.

[2] Y. Hamishebahar, H. Guan, S. So, and J. Jo, "A comprehensive review of deep learning-based crack detection approaches," *Appl. Sci.*, vol. 12, no. 3, p. 1374, Jan. 2022.

[3] Y. Li, T. Bao, B. Xu, X. Shu, Y. Zhou, Y. Du, R. Wang, and K. Zhang, "A deep residual neural network framework with transfer learning for concrete dams patch-level crack classification and weakly-supervised localization," *Measurement*, vol. 188, Jan. 2022, Art. no. 110641.

[4] V. R. Gharehbaghi, H. Kalbkhani, E. Noroozinejad Farsangi, T. Y. Yang, A. Nguyen, S. Mirjalili, and C. Málaga-Chuquitaype, "A novel approach for deterioration and damage identification in building structures based on stockwell-transform and deep convolutional neural network," *J. Structural Integrity Maintenance*, vol. 7, no. 2, pp. 136–150, Apr. 2022.

[5] C.-Z. Dong and F. N. Catbas, "A review of computer vision-based structural health monitoring at local and global levels," *Struct. Health Monitor.*, vol. 20, no. 2, pp. 692–743, 2021.

[6] G. C. Lee, S. Mohan, C. Huang, and B. N. Fard, *A Study of US Bridge Failures (1980–2012)*. Buffalo, NY, USA: MCEER Buffalo, 2013.

[7] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Gener. Comput. Syst.*, vol. 135, pp. 364–381, May 2022.

[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Jan. 2009.

[9] L. Xu, S. Lv, Y. Deng, and X. Li, "A weakly supervised surface defect detection based on convolutional neural network," *IEEE Access*, vol. 8, pp. 42285–42296, 2020.

[10] A. Telea and J. J. Van Wijk, "An augmented fast marching method for computing skeletons and centerlines," Univ. Groningen, Johann Bernoulli Inst. Math. Comput. Sci., 2002.

[11] C. Liu, X. Wang, J. Ni, Y. Cao, and B. Liu, "An edge computing visual system for vegetable categorization," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Orlando, FL, USA, Dec. 2019, pp. 625–632.

[12] D. Witten and G. James, *An Introduction to Statistical Learning With Applications in R*. Berlin, Germany: Springer, 2013.

[13] K. Won and C. Sim, "Automated transverse crack mapping system with optical sensors and big data analytics," *Sensors*, vol. 20, no. 7, p. 1838, Mar. 2020.

[14] D. Joshi, T. P. Singh, and G. Sharma, "Automatic surface crack detection using segmentation-based deep-learning approach," *Eng. Fract. Mech.*, vol. 268, Jun. 2022, Art. no. 108467.

[15] L. Ali, F. Alnajjar, W. Khan, M. A. Serhani, and H. Al Jassmi, "Bibliometric analysis and review of deep learning-based crack detection literature published between 2010 and 2022," *Buildings*, vol. 12, no. 4, p. 432, Apr. 2022.

[16] L. Zeng, E. Li, Z. Zhou, and X. Chen, "Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial Internet of Things," *IEEE Netw.*, vol. 33, no. 5, pp. 96–103, Sep. 2019.

[17] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, Munich, Germany, Sep. 2018, pp. 3–19.

[18] D. Ha and Y. Tang, "Collective intelligence for deep learning: A survey of recent developments," *Collective Intell.*, vol. 1, no. 1, Aug. 2022, Art. no. 263391372211148.
[19] Z. Liu, Y. Cao, Y. Wang, and W. Wang, "Computer vision-based concrete crack detection using U-Net fully convolutional networks," *Autom. Construct.*, vol. 104, pp. 129–139, Aug. 2019.
[20] F. Çağlar and Özgenel, "Concrete crack images for classification," *Mendeley Data, V2*, doi: 10.17632/5y9wdsg2zt.2.
[21] Q. Yang, S. Jiang, J. Chen, and W. Lin, "Crack detection based on ResNet with spatial attention," *Comput. Concrete, Int. J.*, vol. 26, no. 5, pp. 411–420, 2020.
[22] A. Reghukumar and L. J. Anbarasi, "Crack detection in concrete structures using image processing and deep learning," in *Advances in Electrical and Computer Technologies*. Berlin, Germany: Springer, 2021, pp. 211–219.
[23] W. Li, W. Wu, H. Wang, X. Cheng, H. Chen, Z. Zhou, and R. Ding, "Crowd intelligence in AI 2.0 era," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, pp. 15–43, Jan. 2017.
[24] J. Zhao, T. Tiplea, R. Mortier, J. Crowcroft, and L. Wang, "Data analytics service composition and deployment on edge devices," in *Proc. Workshop Big Data Analytics Mach. Learn. Data Commun. Netw.*, Budapest, Hungary, Aug. 2018, pp. 27–32.
[25] A. Guo, A. Jiang, J. Lin, and X. Li, "Data mining algorithms for bridge health monitoring: Kohonen clustering and LSTM prediction approaches," *J. Supercomput.*, vol. 76, no. 2, pp. 932–947, Feb. 2020.
[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[27] C. Tan, N. Uddin, and Y. M. Mohammed, "Deep learning-based crack detection using mask R-CNN technique," in *Proc. 9th Int. Conf. Struct. Health Monitoring Intell. Infrastructure*, St. Louis, MO, USA, 2019, pp. 1–7.
[28] R. C. Gonzalez, *Digital Image Processing*. London, U.K.: Pearson, 2009.
[29] M. Oquab et al., "DINOv2: Learning robust visual features without supervision," 2023, *arXiv:2304.07193*.
[30] G. Borgefors, "Distance transformations in digital images," *Comput. Vis., Graph., Image Process.*, vol. 34, no. 3, pp. 344–371, Jun. 1986.
[31] J. Mendez, K. Bierzynski, M. P. Cuéllar, and D. P. Morales, "Edge intelligence: Concepts, architectures, applications, and future directions," *ACM Trans. Embedded Comput. Syst.*, vol. 21, no. 5, pp. 1–41, Sep. 2022.
[32] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
[33] S. Tuli, N. Basumatary, and R. Buyya, "EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments," in *Proc. 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Nov. 2019, pp. 496–502.
[34] Z. Qiu Lin, A. G. Chung, and A. Wong, "EdgeSpeechNets: Highly efficient deep neural networks for speech recognition on the edge," 2018, *arXiv:1810.08559*.
[35] M. Caron, H. Touvron, I. Misra, H. Jgou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9650–9660.
[36] F. Xie and D. Levinson, "Evaluating the effects of the I-35W bridge collapse on road-users in the twin cities metropolitan region," *Transp. Planning Technol.*, vol. 34, no. 7, pp. 691–703, Oct. 2011.
[37] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. 14th USENIX Symp. Networked Syst. Design Implement.*, Boston, MA, USA, 2017, pp. 629–647.
[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.
[39] *Google Coral Dev Board*. Accessed: Sep. 2023. [Online]. Available: https://coral.ai/products/dev-board/
[40] *Kneron*. [Online]. Available: https://www.kneron.com/
[41] A. Mishra, G. Gangisetti, and D. Khazanchi, "Integrating edge-AI in structural health monitoring domain," 2023, *arXiv:2304.03718*.
[42] X. Cui, Q. Wang, J. Dai, Y. Xue, and Y. Duan, "Intelligent crack detection based on attention mechanism in convolution neural network," *Adv. Struct. Eng.*, vol. 24, no. 9, pp. 1859–1868, 2021.
[43] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2921–2929.
[44] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Seoul, South Korea, Oct. 2019, pp. 3009–3018.
[45] M. Flah, I. Nunez, W. Ben Chaabene, and M. L. Nehdi, "Machine learning algorithms in civil structural health monitoring: A systematic review," *Arch. Comput. Methods Eng.*, vol. 28, no. 4, pp. 2621–2643, 2021.
[46] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–37, Oct. 2021.
[47] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[48] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.
[49] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6824–6835.
[50] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, Apr. 2017.
[51] (2019). *ONNX: Open Neural Network Exchange*. [Online]. Available: https://github.com/onnx/onnx
[52] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, and W. Shi, "OpenEI: An open framework for edge intelligence," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, Jul. 2019, pp. 1840–1851.
[53] A. Mishra, G. Gangisetti, and D. Khazanchi. (2023). *Poster-CEAIC: Cloud-Edge Adaptive Intelligence for Concrete Crack Detection and Quantification*, Lincoln, NE, USA. [Online]. Available: https://bridgingbigdata.github.io/pages/bbd2023agenda.html
[54] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 2704–2713.
[55] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 4820–4828.
[56] Q. Zhang, K. Barri, S. K. Babanajad, and A. H. Alavi, "Real-time detection of cracks on concrete bridge decks using deep learning in the frequency domain," *Engineering*, vol. 7, no. 12, pp. 1786–1796, 2020.
[57] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 857–876, Jun. 2021.
[58] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, Nov. 2020.
[59] X. Chen, S. Yin, D. Song, P. Ouyang, L. Liu, and S. Wei, "Small-footprint keyword spotting with graph convolutional network," in *Proc. IEEE Autom. Speech Recognit. Understand. Workshop (ASRU)*, Dec. 2019, pp. 539–546.
[60] Z. Zhang and C. Sun, "Structural damage identification via physics-guided machine learning: A methodology integrating pattern recognition with finite element model updating," *Struct. Health Monitor.*, vol. 20, no. 4, pp. 1675–1688, Jul. 2021.
[61] Z. Xu, H. Gupta, and U. Ramachandran, "STTR: A system for tracking all vehicles all the time at the edge of the network," in *Proc. 12th ACM Int. Conf. Distrib. Event-Based Syst.*, Hamilton, New Zealand, Jun. 2018, pp. 124–135.
[62] R. A. Gandhi, D. Khazanchi, D. Linzell, B. Ricks, and C. Sim, "The hidden crisis: Developing smart big data pipelines to address grand challenges of bridge infrastructure health in the United States," in *Proc. 15th Int. Conf. Inf. Syst. Crisis Response Manag. (ISCRAM)*, Rochester, NY, USA, 2018, pp. 1016–1021.
[63] Y.-S. Yang, C.-M. Yang, and C.-W. Huang, "Thin crack observation in a reinforced concrete bridge pier test using image processing and analysis," *Adv. Eng. Softw.*, vol. 83, pp. 99–108, May 2015.

[64] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, May 2015.
[65] S. O. Sajedi and X. Liang, "Uncertainty-assisted deep vision structural health monitoring," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 36, no. 2, pp. 126–142, Feb. 2021.
[66] A. Khaloo, D. Lattanzi, K. Cunningham, R. Dell'Andrea, and M. Riley, "Unmanned aerial vehicle inspection of the placer river trail bridge through image-based 3D modelling," *Struct. Infrastruct. Eng.*, vol. 14, no. 1, pp. 124–136, Jan. 2018.
[67] S. Bhowmick, S. Nagarajaiah, and A. Veeraraghavan, "Vision and deep learning-based algorithms to detect and quantify cracks on concrete surfaces from UAV videos," *Sensors*, vol. 20, no. 21, p. 6299, Nov. 2020.
[68] A. Mishra, G. Gangisetti, Y. E. Azam, and D. Khazanchi, "Weakly supervised crack segmentation using crack attention networks (CrANET) on concrete structures," *Struct. Health Monit.*, 2024.
[69] B. Gaurav, Z. Jacobson, S. Madhav, A. Queirolo, and N. Santhanam, "Artificial-intelligence hardware: New opportunities for semiconductor companies," McKinsey, Tech. Rep., 2019. [Online]. Available: https://www.mckinsey.com/industries/semiconductors/our-insights/artificial-intelligence-hardware-new-opportunities-for-semiconductor-companies
[70] R. Singh and S. S. Gill, "Edge AI: A survey," *Internet Things Cyber-Phys. Syst.*, vol. 3, pp. 71–92, Mar. 2023.
[71] E. Badidi, K. Moumane, and F. E. Ghazi, "Opportunities, applications, and challenges of edge-AI enabled video analytics in smart cities: A systematic review," *IEEE Access*, vol. 11, pp. 80543–80572, 2023.
[72] L. Wulfert, J. Kühnel, L. Krupp, J. Viga, C. Wiede, P. Gembaczka, and A. Grabmaier, "AIfES: A next-generation edge AI framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–16, 2024, doi: 10.1109/TPAMI.2024.3355495.

**ANOOP MISHRA** is an Responsible AI researcher and a doctoral candidate at the College of Information, Science & Technology at the University of Nebraska at Omaha. His main research focuses on fairness, transparency, explainability, perception, and human-in-loop in AI/ML systems. His recent focus includes responsible AI practices in Structural Health Monitoring.

**GOPINATH GANGISETTI** is currently pursuing the degree with the University of Nebraska Omaha. He is a Research Assistant with the University of Nebraska Omaha. His research interests include the intersection of cloud-based machine learning applications and software engineering.

**DEEPAK KHAZANCHI** received the B.Tech. degree (Hons.) in civil engineering from the Indian Institute of Technology Kharagpur, the M.B.A. degree from SIU Carbondale, and the Ph.D. degree from Texas Tech University. He holds affiliate appointments at the UNO's International Studies Program and the Goldstein Center for Human Rights. He is currently a Professor of information systems and quantitative analysis and the Executive Director of the Center for Management of IT, University of Nebraska Omaha. He has collaboratively received $10 million in grants/donations/awards from private foundations, corporations, federal, and state agencies to support his teaching/research and community engagement activities. His current research publications are in the following areas: applied AI/ML, perceived fairness of AI/ML systems, fast response virtual teams during the crisis, mhealth interventions, virtual project management, metaverse technology capabilities, B2B risk, and mixed research methods.