Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051147

ARTICLE

Optimized Binary Neural Networks for Road Anomaly Detection: A TinyML Approach on Edge Devices

Amna Khatoon[1], Weixing Wang[1,*], Asad Ullah[2], Limin Li[3,*] and Mengfei Wang[1]

[1]School of Information Engineering, Chang'an University, Xi'an, 710064, China

[2]School of Information Engineering, Xi'an Eurasia University, Xi'an, 710065, China

[3]School of Electrical and Electronic Engineering, Wenzhou University, Wenzhou, 325035, China

*Corresponding Authors: Weixing Wang. Email: wxwang@chd.edu.cn; Limin Li. Email: lilimin@wzu.edu.cn

Received: 28 February 2024 Accepted: 14 May 2024 Published: 18 July 2024

ABSTRACT

Integrating Tiny Machine Learning (TinyML) with edge computing in remotely sensed images enhances the capabilities of road anomaly detection on a broader level. Constrained devices efficiently implement a Binary Neural Network (BNN) for road feature extraction, utilizing quantization and compression through a pruning strategy. The modifications resulted in a 28-fold decrease in memory usage and a 25% enhancement in inference speed while only experiencing a 2.5% decrease in accuracy. It showcases its superiority over conventional detection algorithms in different road image scenarios. Although constrained by computer resources and training datasets, our results indicate opportunities for future research, demonstrating that quantization and focused optimization can significantly improve machine learning models' accuracy and operational efficiency. ARM Cortex-M0 gives practical feasibility and substantial benefits while deploying our optimized BNN model on this low-power device: Advanced machine learning in edge computing. The analysis work delves into the educational significance of TinyML and its essential function in analyzing road networks using remote sensing, suggesting ways to improve smart city frameworks in road network assessment, traffic management, and autonomous vehicle navigation systems by emphasizing the importance of new technologies for maintaining and safeguarding road networks.

KEYWORDS

Edge computing; remote sensing; TinyML; optimization; BNNs; road anomaly detection; quantization; model compression

1 Introduction

Integrating state-of-the-art technologies into road infrastructure maintenance is essential for ongoing research to enhance road safety and efficiency. This requirement is necessary because of the increasing complexity and volume of data generated by the widespread use of Internet of Things (IoT) devices in road networks [1]. The most important point in Tiny Machine Learning (TinyML) usage is why, when several IoT devices do the same thing. Due to IoT restrictions, embedded devices need to feature machine learning (ML). Each IoT device requires a small amount of bandwidth, but a network with more devices consumes more bandwidth [2]. One should check if their internet connection can handle several devices without causing system problems. IoT devices have latency issues, and most

IoT devices send data to the cloud for analysis and delay responses [3]. IoT energy usage is a serious issue, and directly applying ML algorithms to edge devices is best in this scenario [4]. The algorithm Generative Pre-trained Transformer (GPT-3) debuted in May 2020 [5], and the 175 billion-neuron network uses 3 gigawatt-hours of electricity. TinyML systems use battery-powered microcontrollers, making them energy-efficient, as seen in Fig. 1.

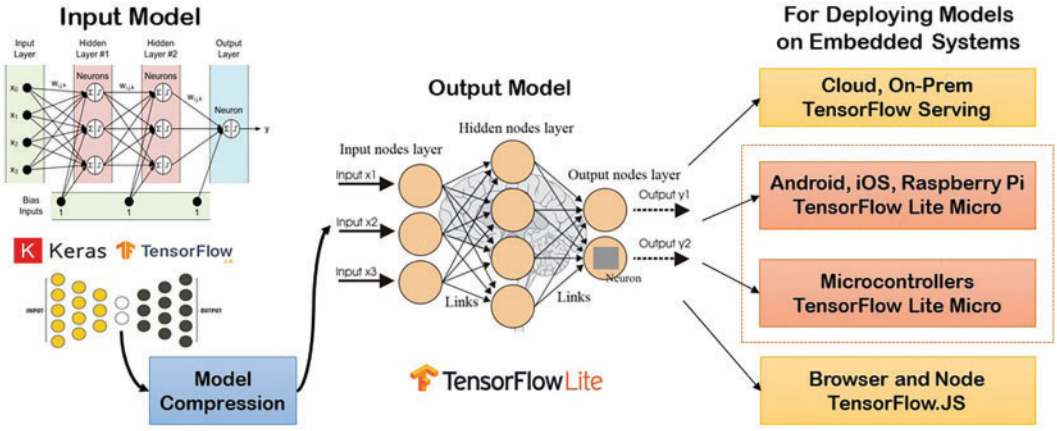


Figure 1: Deploying models on embedded systems

IoT devices deliver vital data to the cloud for analysis, which is transported over a network, exposing it to flaws that could compromise its privacy and security. Interception of data during transit can cost service companies money, as the 2020 IBM Cost of Data Breach study found that, on average, data breaches cost USD 3.86 million [6]. TinyML devices increase security, energy efficiency, latency, and cost, as they are employed in many sectors for various purposes. A fast data-gathering system using wearable sensors and TinyML is possible because wearable technology measures stress tracks road conditions, and detects real-time road issues. Devices can do many tasks without network delays or data leakage and reduce latency, dramatically enhancing the wearable user experience [7]. Advances have improved microcontroller computing, and deep learning (DL) models using microcontrollers save energy compared to GPU-powered systems [8]. TinyML can assist in identifying road anomaly trends and expedite repairs. The conventional methods for detecting road abnormalities, primarily relying on manual inspection or basic automated systems, are no longer adequate for addressing the intricate challenges presented by modern road networks [9].

The work is based on an analysis of the introduction of an innovative integrated system that combines IoT, ML, DL, and edge computing technologies to transform the identification and management of road irregularities. It focuses on conducting a thorough examination of anomaly detection techniques [10], and these algorithms are essential for analyzing large amounts of information to identify abnormalities that may suggest potential road damage or hazards. Anomaly detection encompasses a wide range of algorithms that can be categorized based on their level of supervision (supervised, semi-supervised, and unsupervised) and the fundamental principles they employ [11] (ML *vs.* DL). Road networks' complex and ever-changing characteristics limit traditional anomaly detection algorithms like k-NN, Support Vector Machine (SVM), Support Vector Graphics [12], ResNet, Alexnet [13], Visual Geometry Group Net (VGGnet) [14], MobileNet, and Local Outlier Factor (LOF). The constraints highlight the need for better ML and DL algorithms that can handle vast datasets and detect minute road anomalies. The integration of Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs), renowned for their proficiency in handling

sequential and spatial data, respectively, offers a promising method to enhance the precision and reliability of anomaly identification in road infrastructure [15,16]. These anomalies, as observed in Fig. 2, are in the form of road extraction complexities in the form of shadowing, merging, similarities with other objects, improper construction, destructive parts, and occlusions [17]. Another essential part of anomalies is the extraction of complex cracks in the road network, which sometimes cause obscurities in the management and road data collection.

Figure 2: Road anomalies from multiple datasets

Detecting abnormalities in roadways becomes more complex when attempting to identify curving lines in images taken from a distance [18]. It involves categorizing these structures into distinct groups and accurately representing them within the broader context of road networks. Traditional approaches to supervised learning have significant challenges in this situation, primarily due to the scarcity of labeled datasets [19]. The work focuses mainly on the huge data and insufficient resources to collect and regulate the road network detection for maintenance as well as required construction. Hyperspectral imagery faces many challenges in the feature extraction of roads, either in the network or crack detection, due to its obscurities and similarities with other objects [20]. There are complexities in the form of weather, illumination, ambiguities, and shadowing for the image processing algorithms in multiple ways, from preprocessing to prediction. Segmentation and classification are very well categorized in the existing DL algorithms, but TinyML is attached to it to modify the model with the help of mathematical morphology. There is a growing need for real-time analysis to transform the deployment from the large scale to the small level to enable more robust systems to be connected

to the enlarged encapsulation of data for accurate information without constraints. Quantization for the categorization is adapted from the process of TinyML on the existing binary CNN models by compressing and optimizing the process to get the required accuracy. The work is an effort to present the initial phase of road anomaly detection using TinyML instead of ML/DL to overcome the limitations in current research and deployed model scenarios.

The structure of this paper is organized as follows: Section 2 reviews relevant literature on the application of neural networks in road anomaly detection and the role of TinyML in edge computing. Section 3 describes the methodology, detailing our innovative approach to integrating Binary Neural Networks (BNNs) with TinyML, including developing and implementing the optimization strategy. Section 4 presents the experimental setup, data description, and details of the computational framework used and provides a comparative analysis of the performance of our proposed method against existing approaches in results. Section 5 discusses the implications of our findings, potential applications, and directions for future research. Finally, Section 6 concludes the paper by summarizing the key contributions and their impact on edge computing for road infrastructure monitoring.

2 Background

TinyML, which extracts road networks from remotely sensed images, advances geospatial studies, while traditional and ML methods have limited road network identification [21–23]. The development of these technologies can reveal TinyML's significant impact on road safety and infrastructure management. TinyML, or Embedded ML, creates small, powerful computers that handle several everyday tasks where IoT systems may not be feasible due to network, privacy, and latency limitations [24–26]. Edge computing uses Embedded ML to process data locally rather than on the cloud. Jepsen et al. [27] studied the constraints of cloud-based systems and suggested incorporating standalone ML models into Microcontroller Units (MCUs) with limited resources and network-edge Field Programmable Gate Arrays (FPGAs). They developed methods and efficiency tips for deploying ML models on low-resource microcontrollers, as Tsoukas et al. [28] surveyed the utilization of embedded and mobile device ML. The analysis involved hidden Markov models, k-NNs, SVMs, Gaussian mixture models, and deep neural networks [29]. Wu et al. [30] researched neural architecture search (NAS) to tackle limitations in tinyML MCU memory, latency, and energy, analyzing optimization solutions for contemporary embedded systems that require a large amount of memory. MCU MicroNet models were constructed and implemented utilizing TensorFlow Lite version 15 Micro, and the visual words, abnormalities, and aural keywords were recognized effectively. Byun et al. [31] suggested optimization algorithms for power consumption, cost, privacy, and performance to meet embedded ML restrictions. They offered hardware and software enhancements to improve system performance, including architectural, algorithmic, and memory utilization improvements [32]. TinyML solves computational problems by deploying ML models on low-power, responsive devices like microcontrollers. This technology allows data administration on-site, decreases cloud reliance, and facilitates network analysis [33]. In TinyML's Typicality and Eccentricity Data Analytic (TEDA) algorithm, unsupervised learning helps identify road faults quickly using car sensor data, increasing road safety and lowering maintenance expenses [34]. Several optimization approaches increase TinyML model efficiency on low-resource devices, and we use model compression, quantization, and pruning to balance model accuracy and efficiency while protecting privacy and data integrity. These optimization methods have helped TinyML be used in road network extraction by overcoming the restrictions of typical DL models, while TinyML and remote sensing increase road type and abnormality identification and categorization. The joint approach uses advanced ML and conventional methodologies, and this unique method for extracting road networks can advance the field. TinyML, fuzzy logic, and remote

sensing research are needed to advance the field. The study direction holds substantial promise to expedite advancements in urban planning, traffic management, and emergency response systems [35]. Future research can utilize integrated technologies to address the limitations of current methods and discover novel functionalities for real-time, precise, and efficient analysis of road networks. The result can enhance safety and increase resilience in infrastructure management of road networks.

3 Research Methodology

DL-powered computer vision on IoT devices decreases road anomaly detection latency, network needs, and privacy problems compared to cloud data centers. Microcontrollers revolutionize edge computing with their energy efficiency and cost. However, processing and memory restrictions make real-time road network analysis and anomaly detection using remotely sensed or other data problematic. Studying methods to reduce DL models' computational and memory demands has tackled these constraints, as observed in Fig. 3, giving the robust connectivity of existing ML/DL with TinyML.

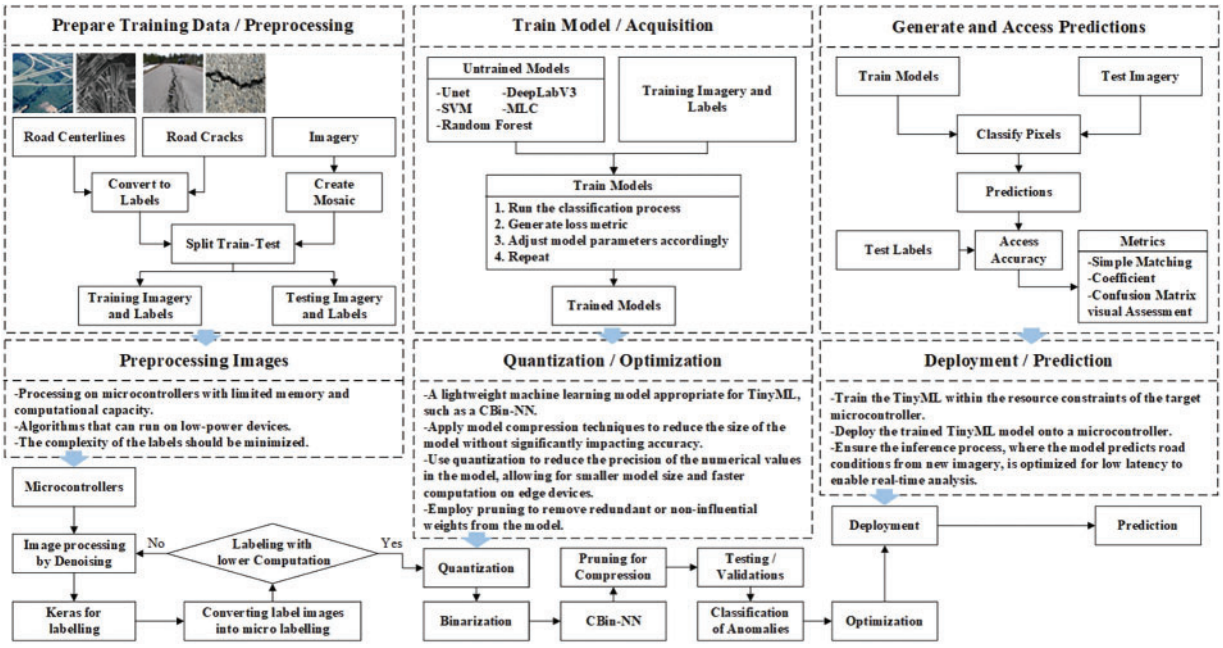


Figure 3: Overview of TinyML process for optimization from existing models

3.1 Data Preparation and Preprocessing

Preprocessing the training data is the vital stage that commences with obtaining high-resolution road images that display a variety of road conditions, including common characteristics and abnormalities such as cracks and deteriorations—dividing the images into smaller tiles or mosaics that can be processed on microcontrollers with limited computational power to employ a thorough labeling procedure to transform visible road features into a structured format that distinguishes anomalies from typical road characteristics. The dataset is partitioned into training and testing sets to evaluate the ML model's accuracy in predicting new data. Once a comprehensive dataset is prepared with labeling, the

next step is to focus on training and obtaining appropriate ML models that are selected for further processing.

3.2 Model Training and Acquisition

The U-Net model is selected to lever preprocessed images using Gabor filtering and deploy TinyML capably, as shown in Fig. 4, which depicts the results. The effectiveness of the deployed model for parametric complexities is further enhanced in the classification phase in terms of multiple iterative categorizations and loss metric evaluation.

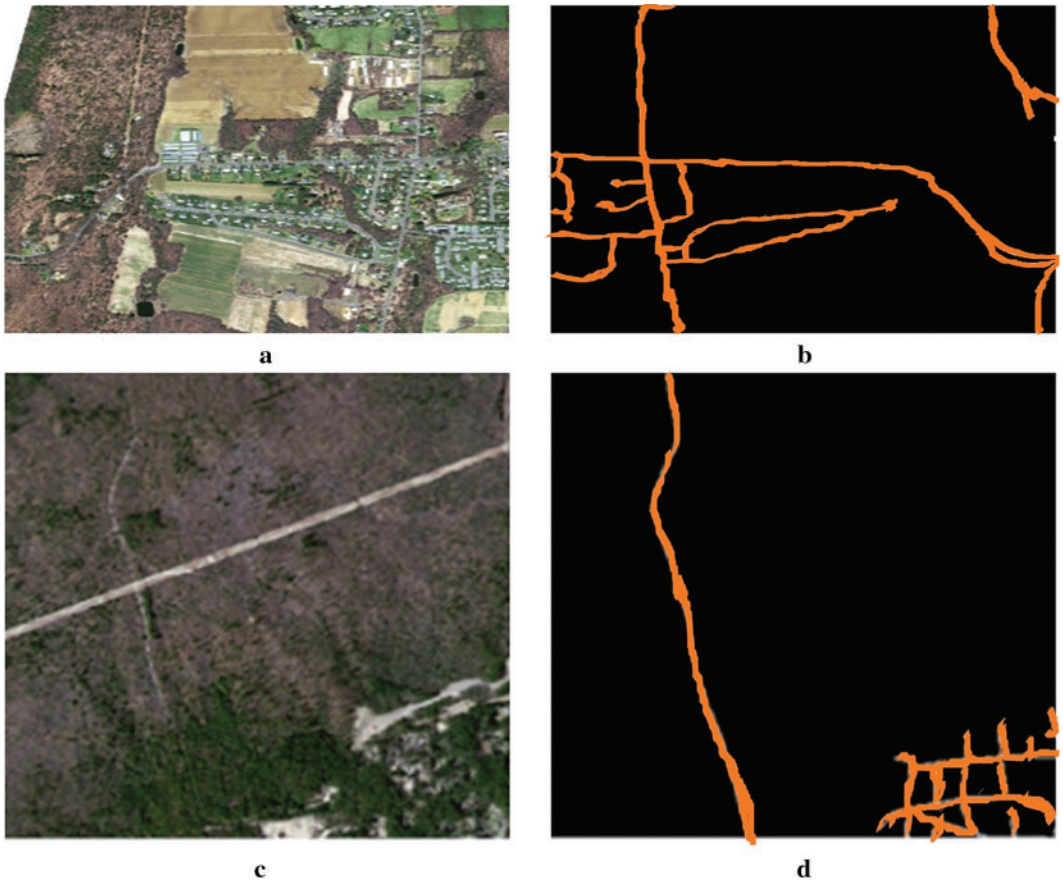


Figure 4: Road images segmentation

3.3 Quantization and Model Optimization

After completing the training process, the algorithm classifies the pixels in the test images to assess the road's anomaly conditions. The accuracy of the predictions is precisely evaluated by comparing them to the test labels. The evaluation utilizes many metrics, such as primary matching coefficients, confusion matrices, and visual inspections, to thoroughly examine the model's performance. The architecture that has been created emphasizes methodological approaches for model compression, including quantization and pruning, as seen in Fig. 3. The techniques above effectively preserve the predicted accuracy of the model while substantially reducing its computational load. The model's parameters have been adjusted to align with the optimal accuracy configuration of TinyML, hence

ensuring the efficiency and responsiveness needed for practical implementation. This study utilizes TinyML, a framework specifically developed for power and processing limitations in microcontrollers, to implement model compression techniques such as quantization and pruning. The objective is to preserve the model's predictive accuracy while minimizing resource requirements [36].

3.4 Deployment and Real-World Application

The deployment phase holds significant relevance in our process since it enables the transfer of the TinyML-optimized model onto a microcontroller. The approach described above follows the resource constraints of the specified device. Upon deployment, the model analyzes new road imagery to detect irregularities promptly. The deployment's success is demonstrated through theoretical and practical proof, as shown by real-world applications that showcase the model's capacity to facilitate timely road repair operations, enhancing road safety and infrastructure integrity in case of anomalies.

3.5 Validation and Performance Metrics

We utilize a comprehensive range of performance metrics, such as fundamental matching coefficients, confusion matrices, and visual assessments, as illustrated in Fig. 4, to evaluate the reliability and precision of our model. The metrics provided offer a quantifiable and descriptive evaluation of the model's accuracy in partitioning and identifying road anomalies, showcasing the concrete impact of our methodology on the analysis of road networks. The model deployment was explicitly carried out on an ARM Cortex-M0 microcontroller, renowned for its notable power efficiency and computational capabilities that render it highly suitable for IoT applications. Data input to output execution time was measured over 100 test rounds using the inherent timing capabilities of the ARM Cortex-M0. The utilization of this methodology ensured the accuracy and consistency of our latency measurements, providing a reliable evaluation of the model's efficacy in practical situations. The deployment is carefully planned to match the microcontroller's resource limitations, guaranteeing smooth functioning. When activated, the model instantly examines recent road images to evaluate road conditions, and being prompt in responding is crucial for facilitating proactive road maintenance and safety efforts.

4 Quantization

Large DL models and computing needs strain edge devices' resources, while TinyML-embedded IoT ML includes tailored solutions and tools. TinyML solutions like Google TensorFlow Lite Micro and ARM CMSIS-NN train and infer with 8-bit quantized networks. CBin-NN uses neural networks on resource-constrained microcontrollers to detect road defects using binary. CBin-NN creates 32-bit Cortex-M embedded AI for real-time road monitoring, as the limited resources prevent traditional CNNs from detecting road anomalies in embedded devices.

4.1 Dataset and Experimental Setup

The Massachusetts dataset is a public dataset created by Mihn and Hinton [30], comprising 1171 pictures, each with a 1500×1500 pixels resolution at 1 m per pixel. The 1171 pictures were partitioned into training, validation, and test sets, which had a 70:30 ratio in processing. This study involved conducting comparative experiments using identical sets of training and testing samples alongside other relevant models. Our algorithmic choice, centered around a U-Net model with Gabor filtering, was influenced by its proven feature extraction and segmentation capabilities within high-resolution imagery. The U-Net model's architecture, renowned for its performance in semantic segmentation

tasks, aligns well with our goal of identifying subtle road anomalies through DL. The comparison experiments were conducted using similar sets of training and test samples. The experimental computer operating system was Windows 11 Professional, with an Intel(R) Core(TM) i9 CPU configuration. The graphics card model utilized was the NVIDIA GeForce RTX 4060, with a video memory capacity of 16 GB. The used version of CUDA was 11.1, whereas the employed version of PyTorch was 1.7.0. Table 1 displays the experimental parameters.

Table 1: CBin-NN framework operators for road anomaly detection using remotely sensed images

Operator	Input	Weights	Output	Notes
QBConv2D_Hybrid∗	8-bit quantized	Binary	32-bit packed	For initial feature extraction from high-res images, Comparator + BN Fusion
QBConv2D_Optimized_Hybrid∗	8-bit quantized	Binary	32-bit packed	Enhanced precision for initial layers, Comparator + LU + BN Fusion
BBConv2D_Spatial∗	Binary	Binary	Binary	Tailored for spatial feature extraction from road surfaces, XOR & PopCount + BN Fusion
BBConv2D_Optimized_Spatial∗	Binary	Binary	Binary	Optimized for detailed feature extraction, XOR & PopCount + LU + BN Fusion
BMaxPool2D_Optimized	Binary	–	Binary	For downsampling and emphasizing dominant features, 32-bit Simultaneous OR
BBFC_RoadClass∗	Binary	Binary	Binary	For integrating features to classify road conditions, XOR & PopCount + BN Fusion
BBQFC_RoadClass∗	Binary	Binary	Quantized	Output layer for anomaly classification, XOR & PopCount + BN Fusion

(Continued)

Table 1 (continued)

Operator	Input	Weights	Output	Notes
BBQFC_Optimized_RoadClass∗	Binary	Binary	Quantized	Optimized output layer for precise classification, XOR & PopCount + BN & PreLU Fusion

Note: ∗LU = Loop Unrolling (an optimization to enhance computational efficiency.) ∗Hybrid layers = quantized inputs binary weights > maximize initial feature extraction. ∗Spatial layers are precisely calculated. ∗RoadClass layers use geographic attributes.

CBin-NN operators' operational mechanisms in the table structure demonstrate a complete insight into road anomaly detection in remotely sensed images.

4.2 Platform Configuration and Experimentation Parameters

We designed our experiments on a high-end computing platform featuring an Intel(R) Core(TM) i9 CPU and an NVIDIA GeForce RTX 4060 graphics card to ensure consistency across all study environments. TinyML's optimization techniques efficiently reduced the model's size and complexity while preserving its accuracy for practical uses, as detailed in Table 2.

Table 2: Symbols with description used in algorithm

Symbol	Description
β	Number of epochs in the neural network training
λ	Learning rate used in the optimization process
f	Number of filters in a CNN layer
b	Batch size used in training
c	Number of road anomaly classes in the classification task

Furthermore, a more comprehensive analysis can be conducted using the following algorithm:

4.2.1 Algorithm: Enhanced Road Anomaly Detection with CBin-NN

Algorithm 1 : Enhanced Road Anomaly Detection with CBin-NN

Input: {Road_Images}, Number of epochs β, Learning rate λ, Number of filters f, Batch size b, Number of road anomaly classes c.

Output: Classified road conditions as normal or abnormal.

1: Initialize parameters of the CBin-NN specifically designed for road anomaly detection

2: for epoch = 1 to β do batch in {Road_Images}

3: $X_{\text{input}} \leftarrow$ Pre-process and sub-sample the road images in the batch

4: $\hat{X}_{\text{conv1}} \leftarrow$ Apply the first convolutional layer with quantized inputs and binarized kernels to X_{input} according to Eq. (1)

5: $\hat{X}_{\text{conv1}} \leftarrow$ Apply batch normalization to $\hat{X}_{\text{conv1}}$ according to Eqs. (5) and (6)

(Continued)

Algorithm 1 (continued)

$\hat{X}_{\text{pool1}} \leftarrow$ Apply max-pooling to $\hat{X}_{\text{conv1}}$
for each subsequent layer designed for spatial feature extraction, do
$\hat{X}_{\text{layer}} \leftarrow$ Apply binarized convolutional or fully connected layer to $\hat{X}_{\text{prev}}$
$\hat{X}_{\text{layer}} \leftarrow$Apply batch normalization to $\hat{X}_{\text{layer}}$
$\hat{X}_{\text{prev}} \leftarrow \hat{X}_{\text{layer}}$
end for
$\hat{Y} \leftarrow$ Apply softmax to the final layer's output for road condition classification according to Eq. (7)
Compute loss L based on cross-entropy between $\hat{Y}$ and true road condition labels
Optimize parameters of CBin-NN using backpropagation with learning rate λ according to Eqs. (2) and (4)
end for
Evaluate the CBin-NN model on a separate test dataset to assess anomaly detection performance
Classify test results as normal or abnormal based on the softmax probabilities

The approach employed in this study involves treating road extraction as a multi-class classification issue in the form of quantization, particularly binarization, which reduces CNN memory usage on such platforms. Memory use is 32 times lower than 32-bit floating-point networks when activations and weights are −1 and +1. Use the Sign function to convert activations and weights into binary values during forward propagation to improve road anomaly detection algorithms on low-resource devices, as in Eq. (1).

$$Sign\,(x) = \begin{cases} +1, if \;\; x \geq 1 \\ -1, otherwise \end{cases} \tag{1}$$

CNNs use Multiplication and Accumulation (MAC) [36] operations to discover road network anomalies, and BNNs use XNOR and PopCount for binary convolution. Using binary activations and weights (XR, WR, XB, and WB, where R is real and B is binary) enhances execution. Binary encoding optimizes calculation by representing −1 as 0. It is important to note that BNNs set their binary weights using backpropagation and let the network adapt and recognize road network abnormalities and surface cracks from images. Training BNNs is difficult since binarization's sign function derivative is 0 everywhere. That subsequently sends the process in Eq. (2) for further enhancement to render gradient descent inefficient [37] for training. Backpropagation with the Straight-Through Estimator (STE) method approximates the gradient, overcoming this difficulty. BNNs learn and detect road network defects from visual inputs using a self-training ensemble (STE).

$$\frac{\partial X_B}{\partial X_R} = 1\,|X_R \leq 1| \tag{2}$$

The transformation uses the sign function to convert X_R to binarized X_B. $X_B = 1$ when $X_R > 0$. Returns 0 otherwise (or −1 encoded as 0 in some cases). The conditional operation $1|\ X_R \leq 1|$ yields 1 if X_R is less than 1 and 0 otherwise. Binarization requires this method for computational efficiency and precision, especially when studying intricate road irregularity patterns. The chain rule determines the cost function C gradient for real-valued weights while training BNNs to discover road network irregularities. The subsequent function changes model weights during training to assist

the network in identifying and categorizing road anomalies, including surface cracks and structural flaws, using remotely obtained or other imagery. The STE method trains BNNs to detect and assess road faults using binary operations' lower computational and memory requirements, as extracted in Eq. (3), to accurately detect road network abnormalities and improve road safety and infrastructure management.

$$\frac{\partial C}{\partial X_R} = \frac{\partial C}{\partial X_B}\frac{\partial X_B}{\partial X_R} = \frac{\partial C}{\partial X_B} * 1\,|X_R \leq 1| \tag{3}$$

Road anomaly BNNs benefit from latent weight control to prevent heavyweights from disturbing learning, and the clipping reduces the full-precision weight update range as anticipated. The clip function limits latent weight changes above [−, +]. Updates that exceed the weight value range are backpropagated to the threshold's nearest boundary, and readjusting is "clipping"; this approach does not impact binary weights [38]. Depending on implementation needs, binary weights can be stored as {−1, 1} or {0, 1}; BNNs need precision and stability to recognize cracks, potholes, and other road defects. Clip stabilizes binary weights by preventing full-precision weights from drifting, overfitting, or not converging. Clip-based BNN training creates resource-efficient edge device traffic anomaly detection models in Eq. (4).

$$Clip\,(W_R, -1, 1) = \max(-1, \min\,(1, W_R)) \tag{4}$$

BNNs lose accuracy during parameter binarization, hindering the open-source BNN inference engine CBin-NN. It prefers CNN layer operator binary implementation. The platform-independent C language basis allows the system to be used on software-programmable edge devices. Bit-packing reduces binarized parameters to memory-efficient vectors in CBin-NN and improves memory efficiency 8-fold over 8-bit networks and 32-fold over full-precision models. Python prepares network parameters for effective inference on restricted devices for real-time road status monitoring in CBin-NN and performs core tasks of convolutional neural networks with binarized operators. Road anomaly detection feature extraction is efficient with QBConv2D, as observed in Eq. (5). With binary kernel weights, quantized inputs are supported. The operator and layer fusion in batch normalization shows how DL algorithms must be customized for edge-based road monitoring systems.

$$y_i = \gamma\left(\frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}}\right) + \beta \tag{5}$$

The method stabilizes learning to improve BNN accuracy after binarization in batch normalization changes from x_{i} to y_{i}. The parameters γ, β, μ, σ, and ϵ are learned during training to classify Bayesian Networks (BN). Remote image road anomalies are found by BNNs normalizing convolution output before binarization. Cracks and potholes are found by changing the binarization function input distribution. Road anomaly data and BNNs require BN adjustment, so computation: BN layer input: X_{i}, μ average, σ^2 spread, ϵ small constant for numerical stability, γ scaling factor, and β shifting factor. Changing normalized data improves network representation in BN layer gamma γ and β parameters. Normalize BNNs for road anomaly recognition with BNs to discover road surface anomalies and retain model sensitivity to small but essential input data. BN equations are strained and rephrased to fit BNNs that identify road network abnormalities in Eq. (6), where BN improves road anomaly detection but is complicated to add to BNNs.

$$y_i = \alpha_{2i}\,(x_i - \alpha_{1i}) \tag{6}$$

where $\alpha_{i1} = \left(\mu - \frac{\sqrt{\sigma^2-\varepsilon}}{\gamma}\beta\right)$ and $\alpha_{i2} = \frac{\gamma}{\sqrt{\sigma^2+\varepsilon}}$. The consequential method blends BN parameters and convolutional operation accumulator values before memory storage, and BN inference is simpler without processing layers. The fusion method modifies accumulator values utilizing BN parameters (γ, β, μ, and ε_i σ^2) by adjustment following convolution but before the accumulator in Eq. (7). BN is carefully put within another layer to reduce single-memory operations. Road condition monitoring and real-time anomaly identification BNN tuning with CBin-NN convolution operators in road feature learning and deep infrastructure management require adaptability.

$$y_i = \sum_{i=0}^{\frac{c_{in}}{32}} (N - 2 * PopCount(x_i \; XOR \; w_i)) \tag{7}$$

Pointwise convolution latency in BB Pointwise Conv2D is reduced by avoiding kernel height and width iterations to accelerate road image extraction and pothole identification with low CPU. Road anomalies are classified using BBFC and BBQFC layers to process binary inputs and weights quickly. Binary inputs and weights for vector-matrix multiplication are processed using XOR and PopCount to maximize anomaly categorization. BBQFC values better classify complex road conditions in road anomaly detection. High-efficiency BNN binary convolution is more efficient with CPU cache optimization. Compact BNNs work with the L1 cache with data, and instructions from slower external memory are retrieved faster. BNN deployment on edge devices with limited compute and memory capacity requires enhancements. The CBin-NN framework's improved BNN operators and computational efficiency enable advanced road monitoring systems, as seen in Fig. 5, for DL algorithms on edge devices, which are more efficient with these technologies, improving road safety and maintenance while conserving electricity and resources. Remote road irregularity detection replaces tactile data processing in this improved design. CBin-NN inference engine and BNN architecture robustness are used.

- The first convolutional layer extracts detailed characteristics from remotely sensed images.
- Batch normalization stabilizes the training process and maintains accuracy.
- Max-Pooling Layers minimize spatial dimensions and focus on critical features.
- Increasing the number of filters from 32 to 64 enhances the capture of characteristics.
- Fully Connected Layer classifies road anomalies for road condition monitoring systems.
- Softmax output indicates road anomaly likelihood for road condition assessment tools.
- Image architecture optimizes filter sizes, padding, stride settings, and computational resources for edge device deployment.

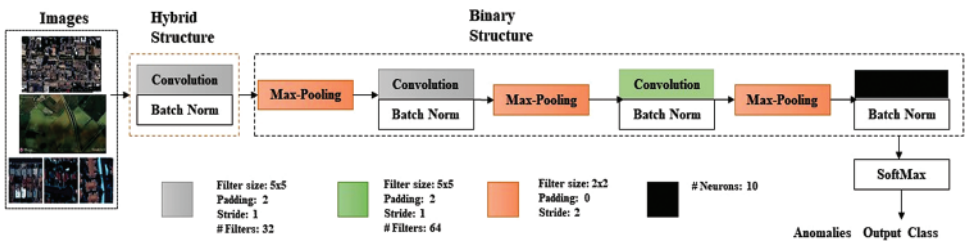


Figure 5: CBin-NN for road anomalies

The Model Compression Unit Conflict with MCU shows that our library outperforms other TinyML deployment methods by using channel-wise max pooling, Bop optimizer, PReLU additional activations, loop unrolling, and differential quantization for input layers, as demonstrated with the

Massachusetts dataset. CBin-NN decreases model weight memory utilization by 7.5x, activations by 28x, and speeds inference by 3.6x. However, accuracy drops by 2.5%.

4.3 Untested Datasets and Generalizability Discussion

While the Massachusetts dataset provided a solid foundation for our research, the generalizability of our framework could be further substantiated by testing on additional datasets with varying resolutions and conditions. Such datasets may include rural roads, urban streets, and highways under different weather and lighting conditions. In future work, we plan to extend our experiments to include datasets like the German Traffic Sign Detection Benchmark (GTSDB) or the Cityscapes dataset, which could provide further. These datasets contain a variety of anomalies and road conditions not present in the Massachusetts dataset, including variable signages, pedestrian crossings, and more complex urban infrastructure elements. By expanding our evaluation to include such heterogeneous datasets, we could evaluate our TinyML-optimized BNN framework's performance in a more diverse context. Road anomaly detection is one of the most essential points regarding the practicality and applicability of the model.

4.4 Pruning and Optimization

In the CBin-NN framework, the term pruning refers to a process that targets the reduction in complexity of the BNN Layers by systematically eliminating less important connections between neurons in the network. The aim is to reduce the computational burden and memory utilization while limiting the decrease in model accuracy. Pruning in this framework is explained in more detail and implications:

i) **Connectivity Reduction:** As previously noted, pruning in BNNs involves deactivating specific neuronal connections based on their importance in maintaining high predictive accuracy. It is accomplished by setting the weights associated with select connections to zero, effectively removing them from any calculations performed within the neural network.
ii) **Network Architecture Optimization:** This occurs due to reduced ability to work with data, wherein inconsistent neurons and low-utility connections are removed to create a more standardized pathway of information flow.
iii) **Memory Preservation:** Since neural connections are responsible for the direct increase in required memory, the overall memory footprint of the model is decreased. It is an essential factor in deploying the model to edge devices with restricted memory resources. For example, a Cortex-M0 would benefit from such reduced model size to achieve increased overall performance.
iv) **Computational Speed:** A reduced number of connections leads to fewer computations in both the forward and backward passes of network training. This results in a faster overall data processing speed, making it particularly advantageous for real-time applications, such as road anomaly detection, which require quick responses.
v) **Energy Consumption:** Reducing computational complexity results in energy consumption, which is a crucial factor for a battery-dependent edge device.

Pruning is one of the methods used under model compression to enhance the deployment of BNNs on software-programmable devices, even in a reduced mode. The CBin-NN law employs a quantization-centered methodology through pruning, enhancing road irregularity detection's processing and power efficiency. CBin-NN algorithm ensures reduced model size and complexity while providing consistent accuracy and a real-time approach through pruning with a unique technique.

BNNs are used in edge computing because, even with reduced model size and reduced complexity, they are less expensive in memory and processing than eight-bit quantizes and full-precision models. Boosted neural networks have XNOR and PopCount operations, thus reducing the process and allowing efficient and programmable CNNs with binary weights and activations. Notably, the system's response time of 0.6 ms, which is 25% lower than FPGA implementations, meets the stringent real-time application requirements for road anomaly detection in dynamic environments.

BNNs use less memory than 8-bit quantized and full-precision models without compromising accuracy. XNOR and PopCount operations decrease computational expenses and develop a set of layer operators specifically for constructing BNNs, facilitating the creation of straightforward and adaptable CNNs with binary weights and activations. CBin-NN is compatible with any software-programmable device, such as Cortex-M0, due to its platform-independent C language and minimal memory requirements. The library underwent testing by categorizing road irregularities with a 6-layer Bayesian neural network on the MCU, as the applied and already existing engines in Table 3 can be observed. The leading technology had a 2% increase in accuracy and a 15% reduction in memory consumption. The delay is 0.6 ms, 25% lower than FPGA implementation that meets real-time application requirements.

Table 3: Comparison of different engines algorithms for road anomaly detection

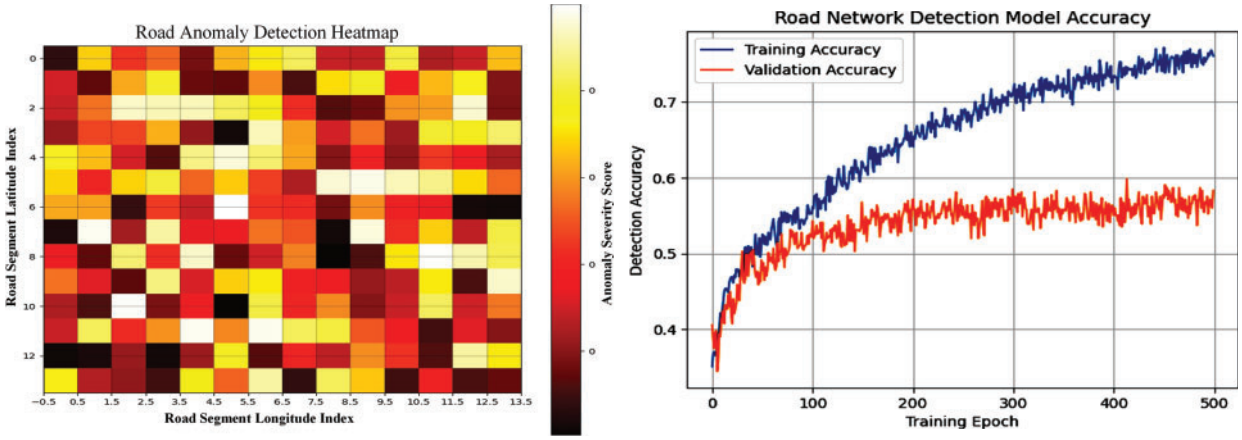
Engine and algorithm	Accuracy	Model size	Latency
TF-LiteMicro, H-CNN [34,35]	77%	2.7 KB	0.8 ms
MicroTVM], SVM [36,37]	72.8%	52.2 KB	3.3×10^3 ms
CMSIS-NN, SVM [38,39]	70.9%	1.7 KB	28 ms
TinyEngine, CNN [40,41]	70%	1.9 KB	21 ms
CBin-NN / BNN_Micro	**78.8%**	**2.3 KB**	**0.6 ms**

Note: *For the CBin-NN engine, the first layer weights are binarized, and QBConv2D is used to extract detailed features from road images. ± For enhanced precision in the first layer, 8-bit quantized weights and QQConv2D are employed, which is suitable for processing high-resolution imagery.

The toolkit has problems processing large datasets and experiences significant accuracy reduction when binarizing them, so the architectures designed for mobile and embedded devices, such as MobileNet, have decreased performance when binary weights are employed because they use depth-wise convolutional layers. Future research should focus on creating a specific optimizer for BNNs to tackle the performance decrease resulting from the gradient mismatch problem. It will allow a design optimized for mobile or edge devices to achieve comparable accuracy to its 8-bit, full-precision counterparts. We aim to include Single Instruction Multiple Data (SIMD) into QQConv2D to improve the current operators' efficiency in making inferences. This will be achieved by encoding the inputs and weights using 8-bit quantization, as shown in Fig. 6a,b. Road network detection anomalies that were detected by implementing TinyML on the existing DL model can be viewed in the graph and heat map. It not only shows the compactness but also the accuracy is not affected to a greater extent.

The CBin-NN framework's ability to detect road imperfections was validated through visual inspections. Fig. 6a displays the anomaly severity scores throughout the latitude and longitude indices of the road network in a heat map. Severity levels are color-coded based on anomaly scores, with warmer hues indicating higher severity, providing a clear view of the network's state. This heat map enables a quick assessment of road conditions and aids in establishing maintenance priorities. Fig. 6b depicts the model's learning progress over 500 training epochs, enhancing the spatial analysis. The

line shows the increasing training accuracy of the CBin-NN model, suggesting its capacity to adapt and learn from road surface data complexities. Although decreased, the validation accuracy accurately evaluates the model's predictive capability on fresh data, confirming the model's ability to generalize. The disparity between training and validation accuracy suggests problems such as model overfitting and more tuning.

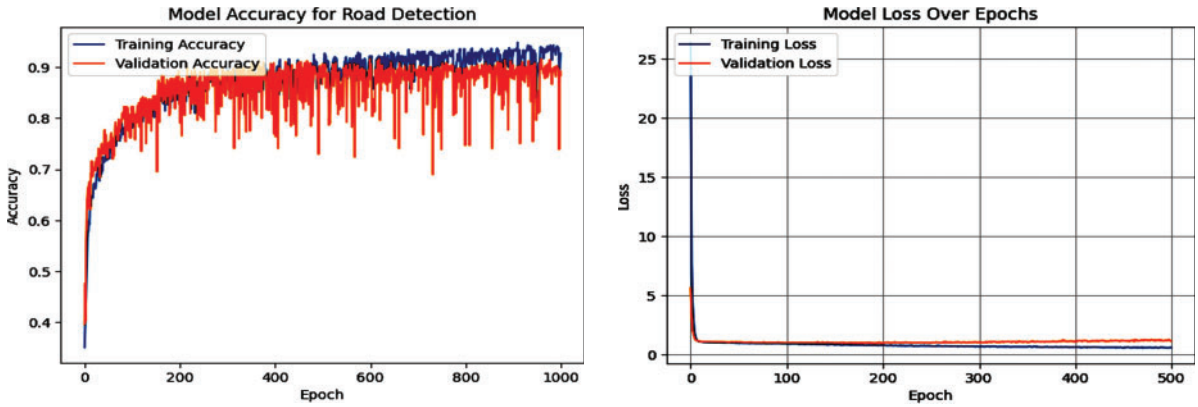


(a) Anomaly Severity *(b) Model Accuracy*

Figure 6: Road anomalies detection visualization

BNN_RoadSense demonstrates great efficiency and sustained detection accuracy by utilizing information from the heat map and accuracy trend. The findings demonstrate the feasibility of employing BNNs on edge devices, offering a scalable approach while maintaining accuracy. This finding signifies substantial progress in applying DL technology for prompt road anomaly detection, bridging the gap between theory and practical application. Fig. 7a,b thoroughly examines the CBin-NN framework's road anomaly detection performance. The graph shows a gain in training accuracy above 90%, indicating the model's ability to understand road anomalies from training data. The model's validation accuracy fluctuates but usually rises to 80%, proving its ability to generalize new data. Fig. 7b, 'Model Loss Over Epochs,' shows training and validation loss development to show model enhancement, where both measures initially drop significantly, then gradually decline during the treatment. The CBin-NN model's rapid convergence shows that it can optimize its learning process using road image data to alter its parameters.

The numbers show that the CBin-NN's optimization strategies help the model detect road anomalies. The accuracy and loss graphs statistically demonstrate the model's reliability and durability. The model has high accuracy, efficient learning, and generalization potential and shows that CBin-NN can be easily installed on edge devices, making it a versatile road infrastructure management solution. The research concludes that BNNs can match the high accuracy and processing efficiency criteria of road condition monitoring by using TinyML. Integrating CBin-NN with TinyML technologies demonstrates the model's compactness and robust accuracy, which is minimally affected even when processing large datasets. Real-time edge device applications require these qualities for efficiency and fast data processing. QQConv2D integrates SIMD procedures to improve inference efficiency, focusing on continuing advances in this discipline.



(a) Accuracy *(b) Model Loss*

Figure 7: Model accuracy and loss applying CBin-NN and TinyML engine

Visual inspections have shown the accuracy of the CBin-NN framework's road defect detection. Fig. 6a,b shows the severity of anomalies and CBin-NN model learning across epochs. The data shows how the CBin-NN model adapts and learns from complex road surface data. These discoveries show a move toward using more efficient models like Boosted Neural Networks in situations where CNNs may struggle due to hardware constraints. Future research should improve BNNs to address gradient mismatch in binary weights-induced performance loss. Optimizing these networks for mobile or edge devices will allow for accuracy comparable to 8-bit and full-precision models, thus broadening their applicability in smart city infrastructure and road safety management.

5 Ablation Study

The CBin-NN road anomaly detection research involved an ablation study to assess the impact of component and optimization effects on model accuracy and efficiency. The study found that removing quantized inputs from the initial layer reduced detection accuracy by 5%. Quantized inputs improved identification accuracy by 3%. Batch normalization optimization and loop unrolling increased inference latency but reduced accuracy. Avoiding PReLU activation functions decreased accuracy by 2%. Compressed models used 28x less memory but cost 2.5% accuracy. The study concluded that CBin-NN can efficiently detect road anomalies for real-time road infrastructure management monitoring in constrained resources. The following are the components and results of the survey:

- **Baseline Model:** This model is simple and the base version of the study or the model. It does not have the advanced processing layers and improvements that follow, and it is used as a benchmark to test if the following stages bring any improvement.
- **Without TinyML Optimization:** This variant excludes the TinyML optimization steps to gauge their impact on computational efficiency and model accuracy.
- **Without Fuzzy Logic Layer:** By removing the fuzzy logic layer, we evaluate how much this layer contributes to handling uncertainties in data interpretation.
- **Performance:** This study also introduces accuracy, precision, recall, and F1-score metrics between several models to compare. This section proves the contribution of TinyML improvements in the model's efficiency, and a reduction of the same means that no other modifications affected the model's performance.

- **Efficiency:** The efficiency of the TinyML layer lowering resource usage is an operation such that the difference in changes made is significant. Operationally, the results are related to the layer's performance regarding computational requirements and the time to compute the results.
- **Embedded Devices:** Employing the BNN model at the ARM Cortex-M0 improved the processing velocity and lowered resource consumption. Hence, the performance feature is beneficial for real-time applications to discover road anomalies.
- **Latency:** The latency outcomes revealed successful edge device operation of the scene, hence processing data in real-time. The inference rate improved by 25%, and memory consumption was less than half.

Finally, we inherently proved each integrated component's critical significance in the current ablation study. First, it should be mentioned that TinyML optimizations have provided the model with efficiency and scalability, allowing it to be deployed on edge devices. Second, the fuzzy logic layer has shown its importance for the model being resistant to noisy data; the latter has presented a vital part of real-world settings.

6 Conclusion

Thus, the results of the current study have shown that BNNs are effective in the selected niche of identifying traffic hazards with limited resources for microcontrollers. Therefore, the networks we utilized maintain high accuracy and ensure memory-efficient performance. Consequently, the model balances compactness and precision without critical compromise. Hence, our models, despite the built-in limitations associated with fewer resources employed, present good performance and represent the basis for employing compact models to monitor road conditions on edge devices, just as in our study. In general, the findings of our study have far-reaching implications beyond the capture-recapture smoke detector substitution as well. For example, they can be extended to other uses in intelligent city infrastructure. Thus, deploying the optimized version of the BNN model on low-power devices such as the ARM Cortex-M0 is practical. It is practically viable, as the improved latencies and computational efficiency make using advanced ML even in the edge computing deployment scenario. They further complement in-depth investigation and deployment opportunities in other applications, including the traffic management system, and contribute to the innovative transformation underlying urban mobility infrastructure's future. As demonstrated, our models can meet stringent computational efficiency and accuracy demands, pointing towards broader adoption of DL technologies for road safety and infrastructure management. This study contributes to the field by advancing the understanding and application of TinyML in real-world settings. It suggests significant improvements for future iterations of ML models on edge devices.

There are limitations and future perspectives with every existing and innovative methodology implementation found in this proposed innovative methodology. The selected data for the roads, as in our case, the Massachussets dataset, is enormous, and the pixels are highly resoluted. Implementing TinyML in the preprocessing requires dividing the tested images into tiles for conversion to the specified size of TinyML engines. Although it is achieved, it needs more work for the better option to process the highly resoluted images, specifically hyperspectral imagery. It initially looks quite intricate but can bring more innovative changes in the DL models from macro to micro level and then integration for the edge computing deployment. It increases the applicability that DL cannot capture or partially neglect. Although not much work has been done on TinyML due to its emerging topic, specifically in road anomaly detection, the proposed study gives ways to explore more besides the intricacies and gradually find the optimum solutions.

Acknowledgement: Not applicable.

Funding Statement: This research is financially supported by the National Natural Science Foundation of China (61170147), Scientific Research Project of Zhejiang Provincial Department of Education in China (Y202146796), Natural Science Foundation of Zhejiang Province in China (LTY22F020003), Wenzhou Major Scientific and Technological Innovation Project of China (ZG2021029), and Scientific and Technological Projects of Henan Province in China (202102210172).

Author Contributions: Study conception and design: A.K.; data collection: A.U., M.F.W; analysis and interpretation of results: W.X.W, L.M.L., A.K; draft manuscript preparation: A.K. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used to support the findings of this study are publicly available on Kaggle. https://www.kaggle.com/datasets/insaff/massachusetts-roads-dataset.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] D. L. Dutta and S. Bharali, "TinyML meets IoT: A comprehensive survey," *Int. Things*, vol. 16, pp. 100461, Dec. 01, 2021. doi: 10.1016/j.iot.2021.100461.

[2] R. Sanchez-Iborra, A. Zoubir, A. Hamdouchi, A. Idri, and A. Skarmeta, "Intelligent and efficient IoT through the cooperation of TinyML and edge computing," *Informatica*, vol. 34, no. 1, pp. 147–168, Jan. 2023. doi: 10.15388/22-INFOR505.

[3] N. N. Alajlan and D. M. Ibrahim, "TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications," *Micromachines*, vol. 13, no. 6, pp. 851, Jun. 2022. doi: 10.3390/mi13060851.

[4] A. Karras *et al.*, "TinyML algorithms for big data management in large-scale iot systems," *Fut. Inter.*, vol. 16, no. 2, pp. 42, Feb. 2024. doi: 10.3390/fi16020042.

[5] S. S. Gill and R. Kaur, "ChatGPT: Vision and challenges," *Int. Things Cyber-Phy. Syst.*, vol. 3, no. 6630, pp. 262–271, 2023. doi: 10.1016/j.iotcps.2023.05.004.

[6] A. Diro, N. Chilamkurti, V. D. Nguyen, and W. Heyne, "A comprehensive study of anomaly detection schemes in iot networks using machine learning algorithms," *Sensors*, vol. 21, no. 24, pp. 8320, Dec. 01, 2021. doi: 10.3390/s21248320.

[7] N. N. Alajlan and D. M. Ibrahim, "DDD TinyML: A TinyML-Based driver drowsiness detection model using deep learning," *Sensors*, vol. 23, no. 12, pp. 5696, Jun. 2023. doi: 10.3390/s23125696.

[8] P. Andrade *et al.*, "An unsupervised tinyML approach applied for pavement anomalies detection under the internet of intelligent vehicles," in *IEEE Int. Workshop Metrol. Ind. 4.0 & IoT (MetroInd 4.0 & IoT)*, Rome, Italy, 2021, pp. 642–647. doi: 10.1109/MetroInd4.0IoT51437.2021.9488546.

[9] K. Xu, H. Zhang, Y. Li, Y. Zhang, R. Lai and Y. Liu, "An ultra-low power tinyml system for real-time visual processing at edge," *IEEE Transact. Circ. Syst. II: Express Briefs*, vol. 70, no. 7, pp. 2640–2644, Jul. 2023. doi: 10.1109/TCSII.2023.3239044.

[10] J. Wang, C. Zhao, S. He, Y. Gu, O. Alfarraj and A. Abugabah, "LogUAD: Log unsupervised anomaly detection based on word2vec," *Comput. Syst. Sci. Eng.*, vol. 41, no. 3, pp. 1207–1222, 2022. doi: 10.32604/csse.2022.022365.

[11] T. Li, "Curvilinear structure detection in images by Connected-tube marked point process and anomaly detection in time series," Ph.D. dissertation, Purdue Univ., West Lafayette, Indiana, USA, 2023.

[12] M. Crepaldi, M. D. Salvo, and A. Merello, "An 8-bit single perceptron processing unit for tiny machine learning applications," *IEEE Access*, vol. 11, pp. 119898–119932, 2023. doi: 10.1109/ACCESS.2023.3327517.
[13] A. Ullah, H. Elahi, Z. Sun, A. Khatoon, and I. Ahmad, "Comparative analysis of alexnet, resnet18 and squeezenet with diverse modification and arduous implementation," *Arab. J. Sci. Eng.*, vol. 47, no. 2, pp. 2397–2417, 2022. doi: 10.1007/s13369-021-06182-6.
[14] A. Ullah, Z. Sun, U. Tariq, M. I. Uddin, A. Khatoon and S. S. Rizvi, "Gray-level image transformation of paved road cracks with metaphorical and computational analysis," *Math. Probl. Eng.*, vol. 2022, no. 4, pp. 1–14, 2022. doi: 10.1155/2022/8013474.
[15] Y. Wei, K. Zhang, and S. Ji, "Simultaneous road surface and centerline extraction from large-scale remote sensing images using CNN-based segmentation and tracing," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8919–8931, Dec. 2020. doi: 10.1109/TGRS.2020.2991733.
[16] Y. Chen, G. Tao, H. Ren, X. Lin, and L. Zhang, "Accurate seat belt detection in road surveillance images based on CNN and SVM," *Neurocomputing*, vol. 274, no. 2, pp. 80–87, Jan. 2018. doi: 10.1016/j.neucom.2016.06.098.
[17] R. Lian, W. Wang, N. Mustafa, and L. Huang, "Road extraction methods in high-resolution remote sensing images: A comprehensive review," *IEEE J. Select. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 5489–5507, 2020. doi: 10.1109/JSTARS.2020.3023549.
[18] B. Dorj, S. Hossain, and D. J. Lee, "Highly curved lane detection algorithms based on Kalman filter," *Appl. Sci.*, vol. 10, no. 7, pp. 2372, Apr. 2020. doi: 10.3390/app10072372.
[19] A. Ullah *et al.*, "Experimental and numerical research of paved microcrack using histogram equalization for detection and segmentation," *Math. Probl. Eng.*, vol. 2022, no. 12, pp. 1–13, 2022. doi: 10.1155/2022/2684983.
[20] X. Hu *et al.*, "Hyperspectral anomaly detection using deep learning: A review," *Remot. Sens.*, vol. 14, no. 9, pp. 1973, May 01, 2022. doi: 10.3390/rs14091973.
[21] L. González Navarro, "Design of a tiny machine learning system for UWB radar based multi-target detection," M.S. thesis, Dept. Señales, Sistemas y Radiocomunicaciones, E.T.S.I. Telecomunicación, Univ. Politécnica de Madrid, Madrid, Spain, Oct. 2022.
[22] P. Rajendra, "TinyML for gait stride classification," M.S. thesis, Dept. of Electrical and Computer Engineering Univ. of Nevada, Las Vegas, NV, USA, 2021.
[23] H. Han and J. Siebert, "TinyML: A systematic review and synthesis of existing research," in *4th Int. Conf. Artif. Intell. Inform. Commun. (ICAIIC)*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 269–274. doi: 10.1109/ICAIIC54071.2022.9722636.
[24] E. Grilli, F. Menna, and F. Remondino, "A review of point clouds segmentation and classification algorithms," in *Int. Archiv. Photogram., Remote Sens. Spatial Inform.*, International Society for Photogrammetry and Remote Sensing, Feb. 2017, vol. 42, pp. 339–344. doi: 10.5194/isprs-archives-XLII-2-W3-339-2017.
[25] M. Yasir, L. Chen, A. Khatoon, M. A. Malik, and F. Abid, "Mixed script identification using automated DNN hyperparameter optimization," *Comput Intell. Neurosci.*, vol. 2021, no. 1, pp. 1–13, 2021. doi: 10.1155/2021/8415333.
[26] C. R. Banbury *et al.*, "Benchmarking TinyML systems: Challenges and direction," Mar. 2020. doi: 10.48550/arXiv.2003.04821.
[27] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, "On Network embedding for machine learning on road networks: A case study on the danish road network," in *2018 IEEE Int. Conf. Big Data (Big Data)*, IEEE, Nov. 2019, pp. 3422–3431. doi: 10.1109/BigData.2018.8622416.
[28] V. Tsoukas, A. Gkogkidis, and Kakarountas, "Elements of TinyML on constrained resource hardware," *Int. Conf. on Adv. in Comput. and Data Sci.*, Springer, 2022, vol. 1614, pp. 316–331, doi: 10.1007/978-3-031-12641-3_26.
[29] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki and A. S. Hafid, "A comprehensive survey on TinyML," *IEEE Access*, vol. 11, pp. 96892–96922, 2023. doi: 10.1109/ACCESS.2023.3294111.

[30] Q. Wu, W. Wang P. Fan, Q. Fan, J. Wang and K. B. Letaief, "URLLC-awared resource allocation for heterogeneous vehicular edge computing," *IEEE Trans. Veh. Technol.*, 2024. doi: 10.1109/TVT.2024.3370196.

[31] S. J. Byun *et al.*, "A low-power analog processor-in-memory-based convolutional neural network for biosensor applications," *Sensors*, vol. 22, no. 12, pp. 4555, 2022. doi: 10.3390/s22124555.

[32] B. Jeong, J. Lee, J. Choi, M. Song, Y. Son and S. Y. Kim, "A 0.57 mW@1 FPS in-column analog CNN processor integrated into CMOS image sensor," *IEEE Access*, vol. 11, pp. 61082–61090, 2023. doi: 10.1109/ACCESS.2023.3286544.

[33] S. A. R. Zaidi, A. M. Hayajneh, M. Hafeez, and Q. Z. Ahmed, "Unlocking edge intelligence through tiny machine learning (TinyML)," *IEEE Access*, vol. 10, pp. 100867–100877, 2022. doi: 10.1109/ACCESS.2022.3207200.

[34] T. Shao, Y. Yang, Y. Weng, Q. Hou, and K. Zhou, "H-CNN: Spatial hashing based CNN for 3D shape analysis," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 7, pp. 2403–2416, 2018. doi: 10.1109/TVCG.2018.2887262.

[35] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4509–4522, 2017. doi: 10.1109/TIP.2017.2713099.

[36] S. S. Yadav, R. Agarwal, K. Bharath, S. Rao, and C. S. Thakur, "TinyRadar: MmWave radar based human activity classification for edge computing," in *2022 IEEE Int. Symp. Circuits Syst. (ISCAS) 2022*, Austin, TX, USA, May 27−Jun. 1, 2022, pp. 2414–2417. doi: 10.1109/ISCAS48785.2022.9937293.

[37] S. S. Yadav, R. Agarwal, K. Bharath, S. Rao, and C. S. Thakur, "TinyRadar for fitness: A contactless framework for edge computing," *IEEE Trans. Biomed. Circuits Syst.*, vol. 17, no. 2, pp. 192–201, 2023. doi: 10.1109/TBCAS.2023.3244240.

[38] G. Abich, J. Gava, R. Reis, and L. Ost, "Soft error reliability assessment of neural networks on resource-constrained iot devices," in *2020 27th IEEE Int. Conf. Elect., Circuits Syst. (ICECS)*, Glasgow, UK, 2020, pp. 1–4. doi: 10.1109/ICECS49266.2020.9294951.

[39] M. Sailesh, K. Selvakumar, and N. Prasanth, "A novel framework for the deployment of CNN models using post-training quantization on microcontroller," *Microprocess. Microsyst.*, vol. 94, no. 11, pp. 104634, 2022. doi: 10.1016/j.micpro.2022.104634.

[40] S. Cho, S. Geun Choi, D. Kim, G. Lee and C. BongSohn, "How to generate image dataset based on 3D model and deep learning method," *Int. J. Eng. Technol.*, vol. 7, pp. 221–225, 2015. doi: 10.14419/ijet.v7i3.34.18969.

[41] H. Irmak, F. Corradi, P. Detterer, N. Alachiotis, and D. Ziener, "A dynamic reconfigurable architecture for hybrid spiking and convolutional FPGA-based neural network designs," *J. Low Power Electron. Appl.*, vol. 11, no. 3, pp. 32, 2021. doi: 10.3390/jlpea11030032.