

AI-Enabled STM32-Based Railway Track Fault Inspection and Real-Time Monitoring Machine

Dr. A. S. Shirkande¹; Ankita Vitthal Kshirsagar²;
Aditya Balasaheb Patil³; Harshad Nana Shinde⁴

^{1,2,3,4}Department of Electronics and Telecommunication Engineering SPVP's S.B. Patil College of Engineering Indapur, Maharashtra, India

Publication Date: 2026/10/21

Abstract: Derailments caused by cracked or fractured rails, shifted track geometry, absent fasteners and intruders on the line remain a serious safety concern, while walking inspections by staff are slow, tiring and hazardous. This review examines the building blocks of a self-driving inspection vehicle controlled by an STM32 microcontroller with embedded artificial intelligence. The vehicle screens the track for eight anomaly types: cracks, fractured rails, obstructions, absent or damaged fittings, geometry misalignment, people or objects on the line, excessive vibration and elevated temperature. Findings are pushed live to a control room, and the vehicle drives itself back to its charging dock once its battery drops below a safe level. We discuss defect categories, sensor choices, neural-network models that fit on Cortex-M parts, energy-aware homing logic and wireless reporting, summarise approaches in tables, and outline unresolved problems and future work.

Keywords: STM32; Railway Inspection; Crack Detection; TinyML; Edge AI; Sensor Fusion; IoT; Autonomous Return; Predictive Maintenance.

How to Cite: Dr. A. S. Shirkande; Ankita Vitthal Kshirsagar; Aditya Balasaheb Patil; Harshad Nana Shinde (2026) AI-Enabled STM32-Based Railway Track Fault Inspection and Real-Time Monitoring Machine. *International Journal of Innovative Science and Research Technology*, 11(10), 1138-1143. <https://doi.org/10.38124/ijisrt/26oct684>

I. INTRODUCTION

The condition of the track is directly tied to railway operational safety. Steel rails endure cyclic wheel loads, heat-induced stresses, corrosion, settling of ballast and occasional tampering. These actions create fatigue cracks beneath the running surface, transverse flaws, failed welds, slack or absent fastenings and drift in gauge [1]. If such flaws remain unnoticed they can grow into fractures that derail trains.

Routine checking today depends on walking patrols, measurement coaches and handheld or trolley-mounted ultrasonic testers. Such checks happen at intervals, cost a great deal of effort, hinge on the alertness of the inspector and expose workers to moving trains. They also leave behind very little continuous data that could guide maintenance planning. Cheaper microcontrollers, miniature MEMS sensors, compact cameras and low-power radios now allow a small robotic carriage to take over part of this work, which is especially valuable where specialised inspection vehicles are not available in sufficient numbers.

STM32 devices, built on ARM Cortex-M cores, suit this job well. They provide hardware floating-point and DSP operations, a dedicated camera port (DCMI), DMA channels, plentiful ADC, I2C, SPI and UART blocks, and deep sleep

modes at modest cost. When paired with compact neuralnetwork runtimes, such a chip can interpret images and sensor streams locally, without relying on a permanent network link [2], [3].

Here we survey the design of an *AI-enabled railway track inspection and live-monitoring vehicle built on STM32*. It recognises track cracks, fractured rails, obstacles, absent or damaged parts, track misalignment, humans or objects on the line, abnormal vibration and abnormal temperature; it sends warnings to a monitoring station; and it returns to its dock by itself when the battery is nearly empty. The paper proceeds through related work, architecture, detection techniques, embedded AI, the homing strategy, communication, a comparison with other approaches, evaluation, and open problems.

II. RAIL DEFECTS AND RELATED WORK

➤ *Classes of Track Faults*

Cannon and co-authors group rail flaws by their origin: rolling-contact fatigue, wear, defects from welding or manufacture, and damage during handling [1]. A moving robot must look beyond metallurgy and watch the entire track surroundings, namely fastenings, sleepers, geometry, foreign items and heat. Table 1 lists the eight fault categories handled by the vehicle.

Table 1 Target Fault Classes and Signatures

Fault	Typical cause / risk	Observable signature
Track crack	Fatigue, squats, head checks; may grow to rail break	Surface line pattern; ultrasonic or eddy-current echo change
Broken rail	Fully developed crack, thermal contraction	Visible gap; continuity loss; vibration spike at joint
Obstacle	Fallen tree, rock, debris, animal	Short-range distance drop; image blob on path
Missing / damaged component	Absent clip, bolt, fishplate; cracked sleeper	Template mismatch in image; missing feature class
Track misalignment	Ballast settlement, buckling, gauge widening	Gauge deviation from 1676 mm (Indian BG) or 1435 mm
Human / object on track	Trespass, cattle, abandoned items	Thermal (PIR / IR) + vision person class
Abnormal vibration	Loose ballast, rail joint defect, corrugation	RMS and spectral peaks above baseline
Abnormal temperature	Sun kink risk, friction, hot axle spots	Rail temperature above safe limit

➤ Existing Inspection Technologies

Ultrasonic and eddy-current methods remain the reference for finding hidden and surface cracks, yet they need contact or close coupling and generally operate slowly. Vision-based methods have advanced quickly: FaghihRoohi et al. trained deep convolutional networks to sort rail surface defects [4], while Gibert et al. used multitask deep learning on track images to find fasteners, ties and surface flaws [5]. Detectors such as YOLO and Faster R-CNN [6], [7] serve well for obstacles and people, although their standard versions are too large for a microcontroller. Inertial measurement using accelerometers on the chassis or axle box reveals geometry faults and joint defects from vibration patterns. Low-cost IoT robots built with Arduino or ESP32 boards usually realise only one or two functions, such as infrared crack sensing or

ultrasonic collision avoidance; very few unite all eight faults with autonomous energy handling in one controller.

➤ Gap Addressed by This Project

Earlier work divides into costly, accurate commercial equipment and cheap prototypes with narrow scope that ignore battery state. The design reviewed here merges multisensor fusion, on-board AI, live reporting and automatic homing in one STM32-class controller to close that gap.

➤ Summary of Representative Studies

Table 2 condenses the main research directions relevant to this project and what each contributes to, or lacks for, a singlecontroller multi-fault machine.

Table 2 Representative Studies and Gaps

Study / area	Contribution	Limitation for this project
Rail defect taxonomy [1]	Defect classes and causes	No sensing or automation scheme
CNN surface-defect detection [4]	Learned features beat hand-crafted ones	GPU-based; surface defects only
Multitask track inspection [5]	Joint fastener, tie and surface-defect analysis	Offline processing of image data
YOLO / Faster R-CNN [6], [7]	Real-time object and person detection	Too heavy for MCU unless compressed
TinyML runtimes [2], [3], [9]	Int8 inference on Cortex-M	Needs task-specific model and data
Edge ML survey [10]	Accuracy–latency–energy trade-offs	General, not rail specific

➤ Motivation

Railway track failures are among the most serious hazards to passenger and freight operations, and early detection of cracks, broken rails, fastener loss, misalignment and intruders is what separates a routine repair from a derailment. Effective inspection therefore needs several capabilities working together: sensing of many fault types, on-board interpretation of the sensor data, dependable communication with a control room, and enough energy management to keep a vehicle working without staff supervision.

Existing studies address these capabilities mostly in isolation. Some concentrate on image-based defect detection [4], [5], others on compact neural-network runtimes for microcontrollers [2], [3], [9], and low-cost robots usually realise only one or two functions. Very little work combines multi-fault sensing, on-device AI, real-time reporting and automatic return to a charging base on a single low-cost

embedded platform. This survey is motivated by that gap. It compares the available approaches, identifies their strengths and limitations, and gathers the findings needed to design an affordable STM32-based inspection machine for Indian railway conditions.

• System Architecture

The vehicle is a four-wheeled carriage that either rides on or is guided by the rails. It is organised in layers: sensors at the top, an STM32 layer for processing and control, an energy and motion layer, and a communication layer that connects to a control-room dashboard.

✓ Controller

A Cortex-M4 or M7 part such as the STM32F4 or STM32H7 (168 to 480 MHz) is appropriate. DMA moves camera frames through the DCMI port with almost no CPU effort, timers produce motor PWM and read encoders, and an RTOS such as FreeRTOS schedules sensing, inference,

networking and safety jobs. Emergency stopping and battery supervision run as top-priority tasks so that they can interrupt image processing at any moment.

be confirmed by a thermal or PIR reading and also by the vision classifier, which cuts false alarms caused by warm stones or shadows.

✓ Sensing Layer

Table 3 names each sensor and the faults it covers. Overlap is intentional: a person on the track, for example, must

Table 3 Sensors, Interfaces and Faults Supported

Sensor	Interface	Detects
OV2640/OV5640 camera	DCMI / DMA	Cracks, missing parts, humans, obstacles
Eddy-current / ultrasonic probe	ADC / SPI	Surface crack, broken rail gap
ADXL345 / MPU6050 IMU	I2C / SPI	Vibration, tilt, joint shocks
MLX90614 IR / DS18B20	I2C / 1-Wire	Rail and ambient temperature
VL53L0X ToF, HC-SR04	I2C / GPIO timer	Obstacle range, gauge distance
PIR sensor	GPIO	Warm body presence
GPS, wheel encoder	UART / timer	Fault location, odometry
INA219, voltage divider	I2C / ADC	Battery voltage, current, SoC

III. FAULT DETECTION METHODS

➤ Track Crack and Broken Rail

Two methods work together. A downward camera under controlled LED light photographs the rail head, and either a small CNN or a classical chain (grey conversion, Gaussian smoothing, Canny or Sobel edges, morphological closing)

➤ Obstacle and Human / Object on Track

A forward time-of-flight or ultrasonic sensor warns within milliseconds and commands a safe stop once the range is shorter than the stopping distance derived from speed and braking. The camera then identifies the object. Pruned and 8bit quantised detectors based on YOLO [6], [14], [15] or MobileNet [8], [13] can recognise a person, with a PIR or infrared reading to back it up. For inanimate items, a twoclass model separates debris from ordinary ballast texture.

➤ Missing or Damaged Components

Fastenings, clips, bolts, fishplates and sleepers are examined by a detector or patch classifier over regions near the rail foot. Gibert et al. showed that fastener and tie condition can be classified dependably from images [5]. On the microcontroller, a compact CNN labels each sleeperspaced window as *present*, *missing* or *damaged*; an encoder pulse fires the capture at sleeper spacing, which keeps computation low.

➤ Track Misalignment

Two sideways ToF or laser range sensors on the chassis measure the distance to each rail; the gauge equals the sensor spacing plus both readings. A reading outside tolerance (for instance ± 10 mm for maintenance advisories) raises a flag. IMU roll gives cross-level (cant) and sideways acceleration points to alignment irregularities. Median and movingaverage filters suppress ballast noise.

➤ Abnormal Vibration and Temperature

The accelerometer is sampled at 1 to 3 kHz through DMA. CMSIS-DSP routines on the STM32 compute RMS, kurtosis and a 256-point FFT, and a threshold or small neural network compares the features with a healthy baseline learned for each segment. A non-contact infrared sensor measures rail temperature; alerts fire when it passes a set limit or climbs unusually fast, which may signal sun-kink risk or frictional heating. Every alert carries a GPS position and a time stamp.

➤ Sensor Fusion and Alarm Logic

Each detector outputs a confidence c_i between 0 and 1. A weighted fusion score is computed per fault class as

$$C = \sum w_i c_i, \text{ with } \sum w_i = 1 \quad (1)$$

And an alarm is raised when C exceeds a class-specific threshold $\square$ for N consecutive frames, which suppresses single-frame noise. Track gauge follows $G = d_L + d_R + s$, where d_L and d_R are the left and right distances to the rails and s is the sensor spacing. The vibration feature is the RMS value $a = \sqrt{(1/n) \sum a^2}$ over a sliding window. For extracts narrow elongated breaks, in the spirit of [4]. An rms $k = v t + v^2 / (2a)$, obstacles, the safe stopping distance is d_{stop} eddy-current or ultrasonic probe adds a signal whose amplitude or echo delay shifts at a flaw. A fractured rail shows up as an abrupt probe change, a visible gap and a vibration jolt when the wheels pass over it. Combining these indications gives more trustworthy alarms.

Where v is speed, t_r the reaction time of the sensing and control loop and a_b the braking deceleration; the forward sensor range must exceed d_{stop} with margin. Faults are graded into three severity levels: *advisory* (log only), *warning* (alert and continue) and *critical* (stop and alert), as in Table 4 (values are configurable examples, to be tuned on site).

Table 4 Example Severity Grading (Configurable)

Fault	Advisory	Warning	Critical (stop)
Crack	Faint line	Clear line	Gap / rail break

Obstacle / human	—	Beyond d_{stop}	Within d_{stop}
Gauge deviation	5–10 mm	10–20 mm	> 20 mm
Vibration RMS	1.5× baseline	2× baseline	3× baseline
Rail temperature	Near limit	At limit	Above limit

➤ Embedded AI on STM32

Neural networks on Cortex-M chips demand careful model design. Integer quantisation, pruning and operator fusion cut memory use and latency [11], [12], and libraries such as CMSIS-NN [9], TensorFlow Lite Micro [2] and ST X-CUBEAI supply tuned kernels. MLPerf Tiny [3] demonstrates that image classification and anomaly detection fit within a few hundred kilobytes, and the edge-ML survey [10] stresses the compromise between accuracy, delay and energy that guides model selection.

A practical pipeline has two stages. Stage 1 applies cheap rules and statistics to every sample (range limits, vibration RMS, temperature bounds). Stage 2 wakes the CNN only when Stage 1 or the encoder points to a region of interest, which saves energy. Typical models are a MobileNet-style classifier of 100 to 300 kB for crack and component patches, a tiny person detector and a 1-D CNN or autoencoder for vibration anomalies. Training data are rail images and vibration logs expanded with brightness, blur and rotation changes to survive outdoor light. If memory runs short, the STM32 can forward compressed regions to an edge gateway for heavier inference.

Table 5. Embedded Models (Indicative Design Ranges)

Task	Model / method	Approx. footprint	Reference
Crack / surface defect	Small CNN, edge filters	80–300 kB flash	[4], [9]
Missing fastener / sleeper	Patch CNN (int8)	100–250 kB	[5], [8]
Human / obstacle	Tiny YOLO / MobileNet	0.5–2 MB (H7 class)	[6], [8]
Vibration anomaly	FFT + 1-D CNN / autoencoder	10–50 kB	[3], [10]
Misalignment, temperature	Thresholds + filtering	< 5 kB	—

➤ Automatic Return-to-Base on Low Battery

Energy autonomy separates a deployable robot from a lab demonstration. The vehicle tracks its battery state of charge (SoC) by combining voltage, integrated current

(coulomb counting) and a voltage-to-SoC table, because voltage alone is misleading under motor load. Fig. 1 shows the supervising state machine.

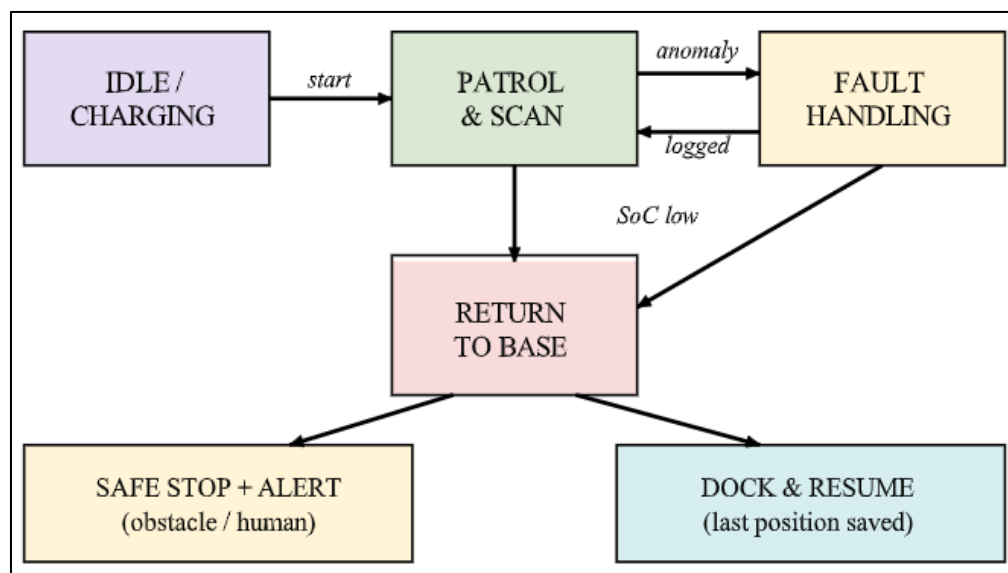


Fig 1 Supervisory State Machine Including Low-Battery Return.

A fixed SoC threshold is not enough, since the energy needed depends on how far the vehicle is from its dock. The controller therefore tests a distance-aware condition:

• Control Algorithm

- ✓ Every 100 ms: read battery V, I; update SoC
- ✓ Compute E_{avail} , d_{base} from odometry 3. if $E_{avail} \leq k \cdot e_m \cdot d_{base} + E_{res}$ or $SoC < limit$:

- ✓ Save position; state = RETURN
- ✓ Else run sensor tasks; fuse; grade severity
- ✓ If critical: stop, alert; wait for clearance
- ✓ In RETURN: slow down, keep obstacle sensing,
- ✓ Retrace path; dock; state = CHARGING
- ✓ When $SoC > resume\ limit$: go to saved position

- **Power Budget**

Table 6 gives an indicative power budget used for sizing the battery. Motors dominate consumption, so mission range

is mostly set by vehicle mass, speed and rolling resistance, and runtime is approximated by $t = (E_{\text{batt}} \times \eta \times \text{DoD}) / P_{\text{avg}}$.

Table 6 Indicative Power Budget

Subsystem	Indicative power
STM32 MCU (active)	0.1–0.5 W
Camera + LED lighting	0.3–1.0 W
Sensors (IMU, ToF, IR, probe)	0.1–0.4 W
Wireless (ESP32 / LoRa / GSM)	0.2–1.5 W (burst)
Drive motors	5–20 W (load dependent)

➤ **Real-Time Monitoring and Communication**

Each event is packed into a short message holding fault type, confidence, GPS position, time stamp, a sensor snapshot and, optionally, a cropped image. Mobile coverage (GSM/4G) is common along rural lines but not guaranteed; LoRa reaches several kilometres at very low power, and Wi-Fi works near stations. Using several links with store-and-forward buffering on an SD card prevents alerts from being lost in dead zones. MQTT, a light publish–subscribe protocol, feeds the dashboard, which shows a live map, severity colours, trend graphs and history for predictive maintenance. A local buzzer and LEDs warn nearby staff at once. Authentication and TLS encryption block forged alerts, which matters for safety-critical infrastructure.

IV. TESTING AND EVALUATION METHODOLOGY

$$E_{\text{avail}} \leq k \cdot e_m \cdot d_{\text{base}} + E_{\text{res}} \quad (2)$$

Here E_{avail} is the usable energy left, e_m the measured energy per metre, d_{base} the distance to the dock from odometry, k a safety factor (usually 1.2 to 1.5) and E_{res} a fixed Validation should move through three levels. Bench tests compare each sensor with a reference instrument (calliper or gauge for alignment, infrared thermometer for temperature, reserve. A soft level (for example SoC under 30%) notifies the operator, while a hard limit (15 to 20%) forces the trip home

regardless. When it triggers, the vehicle records its last inspected GPS and encoder position, finishes or abandons any fault confirmation, turns around and follows its own odometry back, correcting drift with GPS and optional RFID or marker tags at the dock. On the way it keeps obstacle sensing running and lowers speed and non-essential loads such as lighting and telemetry rate to save energy.

At the dock, contact or inductive connectors engage, the state returns to *CHARGING* and logs are uploaded. After recharging, the vehicle resumes from the stored position so that no stretch is skipped. A gap between the return and resume thresholds (hysteresis) prevents repeated switching. Solar-assisted charging, proposed for several low-cost rail robots, could stretch range further.

Shaker table for vibration). Test-track trials use a short rail section with artificial faults, such as machined notches, removed clips, placed obstacles and mannequins, under different light and weather. Field trials follow with railway permission during track possessions. For every fault class the study should report precision = $TP/(TP+FP)$, recall = $TP/(TP+FN)$ and $F1 = 2PR/(P+R)$, plus false alarms per kilometre, detection delay, position error against surveyed marks, message delivery ratio and range per charge. For homing, the main measures are how often the vehicle reaches and docks from various distances and loads, and how much energy remains on arrival. Since a missed critical fault costs more than a false alarm, thresholds for critical classes favour recall, and a human operator confirms every alert before maintenance is ordered.

➤ **Comparative Analysis**

Table 7 Comparison with Existing Approaches

Approach	Faults covered	Cost	On-device AI	Auto return
Manual patrol	Visual only	Labour	No	N/A
Ultrasonic trolley / car	Internal cracks	High	Limited	No
Arduino / ESP32 robots	1–3 faults	Low	Rare	Rare
Drone / vision-only	Surface, obstacles	Med.	Cloud / GPU	Partial
STM32 AI machine (reviewed)	8 fault classes	Low–M ed.	Yes (TinyML)	Yes

The reviewed design gives up the raw accuracy of industrial ultrasonic cars in return for wide coverage, round-the-clock operation, low cost and autonomy. It is best seen as a *screening* aid that marks suspicious spots for confirmation by certified equipment, not as a substitute. The values in Tables 5 and 7 are indicative design ranges taken from the

cited work and typical datasheets; real-track validation is needed to establish measured accuracy, falsealarm rate and range per charge.

V. CHALLENGES AND FUTURE DIRECTIONS

➤ Deployment Scenarios and Benefits

Suitable uses include night or low-traffic patrols on branch lines, checks of embankments and ballast after the monsoon, yards and sidings, and sections prone to accidents such as curves, bridges and level crossings. Gains include less exposure of staff to live-line danger, more frequent inspection, objective time-stamped records, and early warnings that support condition-based maintenance instead of fixed schedules. A prototype bill of materials (STM32 board, camera, IMU, ToF and IR sensors, GPS, wireless module, motors, chassis, battery) should cost far less than commercial inspection vehicles, although certified production units would cost more.

➤ Open Challenges

Limited data. Few public datasets cover Indian railway defects; synthetic augmentation and federated learning across robots look promising. Weather and light. Glare, rain, dust and vegetation hurt vision, which favours multispectral and thermal imaging with sensor fusion. Compute limits. Finer crack detection stretches a microcontroller; STM32 parts with NPUs, distillation and architecture search may help. Safety and approval. The vehicle must never foul a live line, so possessions, train-detection interlocks and fail-safe brakes are essential. Energy. Better SoC estimation, solar charging and swappable packs would lengthen missions. Localisation. GPS drifts near tunnels and cuttings, so visual or RFID landmarks are needed. Future directions include swarms of cooperating units, digital-twin links for predictive maintenance and over-the-air model updates.

VI. CONCLUSION

This review covered the technology behind an AI-enabled, STM32-based track inspection and monitoring vehicle that detects cracks, fractured rails, obstacles, absent or damaged parts, misalignment, people or objects on the line, abnormal vibration and abnormal temperature, and docks itself when its battery runs low. Sensor fusion, a two-stage embedded-AI pipeline, distance-aware homing and resilient wireless reporting together give an affordable and scalable way to screen track often. With enough data and field trials, such robots can supplement standard inspection, lower human exposure to danger and enable predictive maintenance, improving railway safety.

REFERENCES

- [1.] D. F. Cannon, K.-O. Edel, S. L. Grassie, and K. Sawley, "Rail defects: an overview," *Fatigue Fract. Eng. Mater. Struct.*, vol. 26, no. 10, pp. 865–886, 2003.
- [2.] R. David *et al.*, "TensorFlow Lite Micro: Embedded machine learning on TinyML systems," in *Proc. MLSys*, 2021.
- [3.] C. Banbury *et al.*, "MLPerf Tiny benchmark," in *Proc. NeurIPS Datasets and Benchmarks*, 2021.
- [4.] S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, and B. De Schutter, "Deep convolutional neural

networks for detection of rail surface defects," in *Proc. IJCNN*, 2016, pp. 2584–2589.

- [5.] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 153–164, 2017.
- [6.] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE CVPR*, 2016, pp. 779–788.
- [7.] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015.
- [8.] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv:1704.04861, 2017.
- [9.] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs," arXiv:1801.06601, 2018.
- [10.] M. G. S. Murshed *et al.*, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, 2021.
- [11.] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE CVPR*, 2018, pp. 2704–2713.
- [12.] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. ICLR*, 2016.
- [13.] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C.
- [14.] Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE CVPR*, 2018, pp. 4510–4520.
- [15.] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv:1804.02767, 2018.
- [16.] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv:2004.10934, 2020.