IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 October 2025, accepted 3 December 2025, date of publication 8 December 2025,
date of current version 19 December 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3641380

RESEARCH ARTICLE

RATTLE: A Framework for Train Characterization and Identification Through Audio Fingerprinting

LEONARDO CIABATTINI[1], ALFONSO ESPOSITO[1], (Graduate Student Member, IEEE), LUCA SCIULLO[1,2], IVAN ZYRIANOFF[1,2], AND MARCO DI FELICE[1,2]

[1]Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy
[2]Advanced Research Center on Electronic Systems, Bologna, 40123 Bologna, Italy

Corresponding author: Leonardo Ciabattini (leonardo.ciabattini@unibo.it)

ABSTRACT Train characteristics identification serves a vital role in monitoring the traffic on railway infrastructures and correlating it with their structural response. It traditionally relies on camera-based systems, which raise privacy concerns, have high implementation costs, and perform variably depending on visibility conditions. To address these limitations, we present RATTLE, an IoT framework that uses audio fingerprinting to identify trains, count carriages, and estimate transit speed. Our system combines edge devices for data collection, feature extraction, and processing with cloud infrastructure for aggregation and model training. To validate our approach, we collected a comprehensive dataset of video and audio samples, annotated it with ground truth labels, and used it for evaluation in our experiments. We are also making this dataset openly available to the research community to support further advancements in the field. The framework employs a custom Convolutional Neural Network for processing mel-spectrograms, achieving 96% accuracy in train identification, with a mean absolute error of 0.85 for carriage counting and 1.8 m/s for speed estimation. These results were obtained using only 540 training samples per location, with the model requiring less than 1.63 MB of storage, making it suitable for deployment on resource-limited devices.

INDEX TERMS Audio fingerprinting, deep learning (DL), edge computing, Internet of Things (IoT), machine learning (ML).

I. INTRODUCTION

Audio-based vehicle fingerprinting involves identifying vehicles' characteristics, such as presence, model, and speed, using audio signals. Although this approach is less explored compared to computer vision-based techniques, it offers reduced costs and enhanced privacy, particularly when deployed in public, large-scale environments such as roadways or railways. In smart city scenarios, this enables the detection of abnormalities of interest for forensic purposes (e.g., car crashes) [5], [18], [32]. Similarly, applications related to the security of unmanned vehicles have also been explored in the literature [36].

In this study, we explore a novel use case: the design and development of efficient audio-based systems for the identification of train characteristics (TCI). This task involves determining the key properties of a passing train, such as its model, number of carriages, and speed at specific locations along the railway network. The ability to extract such information is essential for achieving two complementary objectives. First, it plays a critical role in Structural Health Monitoring (SHM) of railway infrastructures. Indeed, SHM is an emerging interdisciplinary field at the intersection of civil engineering, the Internet of Things (IoT), and data science [16]. It focuses on assessing the integrity of buildings and infrastructures while enabling predictive maintenance operations through a variety of sensing technologies and processing methods [2]. In case of SHM for railway systems, identifying the train model and the number of carriages facilitates the estimation of the vehicle's weight [1], [22] and hence can be correlated with the structural response of the infrastructure, as recorded by on-site IoT sensors. Through such integration, comprehensive and effective SHM systems capable of monitoring both static and dynamic

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

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

parameters can be deployed [6], [21]. Secondly, a TCI system facilitates the creation of datasets augmented with contextual, transportation-aware annotations. The availability of reliable, real-world datasets is crucial to support Machine Learning (ML)-based applications for SHM, which are still in their infancy despite the promising results [39].

Although TCI tasks can be implemented through camera and computer vision techniques, a significant contribution of our study lies in exploring the feasibility of accomplishing them solely through audio fingerprinting [8]. This approach offers several advantages over camera-based solutions, including enhanced portability, cost reduction, privacy awareness, and increased operativeness, especially in low-light and averse weather conditions [33]. Moreover, it can be more precise than querying static timetables, which are subject to delays or unforeseen changes, and more reliable than laser-based solutions, which can be affected by adverse weather conditions. At the same time, audio-based TCI solutions present novel scientific challenges that must be addressed for effective system design and development. Firstly, suitable datasets must be collected, and appropriate Deep Learning (DL) solutions need to be designed and trained over audio samples. These DL models must balance the trade-off between performance and complexity, as the TCI system is likely intended to operate on edge devices located near railways, which have limited computational capabilities. The TCI system must account for the uniqueness of location in the data processing, both at short and large scales. At the short scale, audio signals sampled at different locations may exhibit variations in amplitude and frequency due to factors such as the position of the edge recording system, the presence of obstacles, and ambient noise levels. At the large scale, the diversity of train models across different geographical locations adds another layer of complexity. Since training a location-specific DL model for every site may not be feasible due to scalability constraints and the limited resources of edge devices, it becomes essential to explore solutions that enable data sharing across multiple locations while efficiently distributing computational tasks across the continuum [19], [24].

This work is motivated by the following research objectives: can we create a railroad monitoring framework that is (i) resilient to different weather and lighting conditions (day and night), (ii) inexpensive and edge-friendly, (iii) privacy-aware, (iv) and capable of accurately characterizing multiple train features (such as train type, carriage count, and speed)?

To address this question, we propose and evaluate the RATTLE framework (Railway Acoustic for Train Tracking and LEarning), an audio-based approach for TCI, aiming to overcome the limitations of existing systems while supporting the needs of SHM and other railway applications. The framework integrates IoT field systems, designed to be deployed near railways, with cloud components for data aggregation and DL model training. A preliminary version of this framework was introduced in [10]. Here, we significantly extend its capabilities to support three key identification tasks from audio signals: train class, train length (in terms of the number of carriages), and train speed. Additionally, we tackle critical issues such as scalability, resource allocation, and location uniqueness by enabling RATTLE to seamlessly operate across the computing continuum. Specifically, we investigated methods for cloud-based data aggregation and generic DL model training by combining heterogeneous datasets from multiple IoT field systems. Simultaneously, we explored strategies for fine-tuning large DL models to the specific characteristics of each edge deployment. This approach minimizes the re-training effort at each location while mitigating computational overhead on edge devices. To support our research, we collected a comprehensive dataset of video and audio samples from passing trains, which we meticulously annotated with ground truth labels for train model, carriage count, and speed. This dataset was used in Section VI. Furthermore, we are committed to fostering research in this domain by making this dataset openly available to the research community.

Our results demonstrate that these requirements can be met effectively. First, our Convolutional Neural Network (CNN) architecture trained on mel-spectrograms matches in accuracy performance (98.5%) the best image classifiers, such as the Vision Transformer [14] at a fraction of the weight (1.63 MB vs 329.5 MB), hence justifying the idea of using audio fingerprinting for TCI tasks. Second, by training on the global dataset and fine-tuning with only 540 samples collected at edge subsystems, we achieved up to 96% accuracy in train classification. The size of the custom CNN model was well within the resource constraints of our IoT field system devices (Raspberry Pi 3 Model B), enabling real-time inference directly on the edge. Finally, using the same number of samples for fine tuning, we obtained significant outcomes for carriage counting (0.85 mean absolute error) and train speed estimation (1.8 m/s mean absolute error).

To summarize, the main contributions of this paper are as follows:

- Design of a continuum-friendly architecture that enables DL-based TCI tasks.
- Development of a prototype for the RATTLE IoT field system, supporting audio recording during train passages, data transmission, and edge-based DL inference.
- Collection, annotation, and open publication of a dataset comprising video and audio samples for TCI.
- Creation of custom DL techniques for train classification, carriage counting, and train speed estimation.
- Comparative analysis of various DL video-based classification techniques (used as baselines) alongside traditional ML and DL audio-based classification methods across multiple datasets.
- Exploration of strategies for fine-tuning a general DL model to specific edge locations, including an evaluation of the sample size required for effective adaptation at each location.

The rest of the paper is organized as follows: Section II outlines the related works in the area, Section III illustrates

the architecture of the whole pipeline, including how we developed the IoT field system for data collection, Section IV describes the datasets utilized in the experiments, Section V details the model architecture, in specific the one utilized for train classification, carriage counting and, speed estimation, Section VI presents the results, and finally Section VII concludes the paper.

II. RELATED WORKS

Audio classification involves mapping audio samples to their respective labels [8]. Several AI techniques have been proposed for this task [37]. On the one hand, traditional methods involve performing classification on extracted features from the audio signal. For instance, Tsalera et al. [34] employed KNN to classify environmental noise that impacts negatively in urban scenarios (e.g., drilling), achieving accuracy performances between 70% and 85%. On the other hand, more recent methods utilize DL architectures originally developed for computer vision applications, adapted for processing mel-spectrograms. The authors of [17] employed the MobileNet [31] to identify birds based on the mel-spectrograms, reaching varying accuracy from 80% (2 classes) to 40% (10 classes). In [38], a cough recognition method to identify COVID-19 clinical cases was proposed. The authors employed a pipeline involving data acquisition, pre-processing to ensure consistent data length, and a CNN-based technique to classify the cough events from mel-spectrograms, achieving more than 90% accuracy with a model of about 1.35 million parameters (5.13 MB). This line of work supports the general effectiveness of mel-spectrogram features paired with compact CNNs for acoustic event recognition in diverse domains; our study adapts this methodological insight to train-centric identification tasks under railway constraints rather than clinical screening. Following the breakthrough of the vision transformer [14], in [8] the authors designed a transformer for audio classification, introducing a hierarchical structure to reduce the model size and training time.

In parallel, recent advances in audio fingerprinting have specifically targeted robustness against background noise and distortion in open environments. While such works are not directly tailored to train characterization, they reinforce the viability of sound-based identification when environmental conditions are challenging—a property beneficial to our application domain. In particular, Kamuni et al. [20] focus on peaks over mel-spectrograms to filter out background noise, while we make use of the average energy of the background noise to identify our detection window. Ultimately, Kamuni's approach could be integrated into our data processing pipeline, potentially improving the model's performance.

The DL techniques mentioned above often demand computational resources that are typically unavailable in IoT environments. Optimizing classification models for constraint environments necessitates finding a balance between efficiency, accuracy, and performance [27]. To this purpose, Mou and Milanova [27] applied compression techniques as pruning and quantization to CNN models for audio classification. Malik et al. [25] implemented non-speech audio recognition at the edge utilizing a CNN model, which was compared with MobileNet. Moreover, they evaluated the inference and loading times on constrained edge devices, demonstrating its feasibility. In [26], the performance of DL algorithms for audio classification was investigated on edge devices, considering various compression techniques. Particular attention was given to XNOR networks [30], where all layers, except the first and last, are binary.

Regarding domain-related works, it is worth highlighting that most of TCI solutions proposed in the literature leverage cameras as sensing technologies. A survey related to the usage of acoustic and visual sensors in railway maintenance applications can be found in [13]. In [22] authors proposed a solution for counting and identifying train carriages using computer vision and DL algorithms. Carriage number recognition through the ResNet DL architecture was investigated in [35]. Finally, outside the audio modality, cost-effective weigh-in-motion (WIM) systems based on wireless vibration sensing have been shown for road traffic [3]. Although orthogonal to our current scope, they motivate a multimodal future extension wherein audio-based TCI can be complemented with vibration to infer axle/gross weights in railway scenarios. Vehicle tracking systems based on audio fingerprinting have received relatively little attention in the literature, with most studies focusing primarily on cars or Unmanned Aerial Vehicles (UAVs). The study in [18] explored the feasibility of real-time audio event detection for forensic applications across various scenarios, including car crash incidents. Mel-spectrogram-based features were used for vehicle speed estimation in [5]. The method was further expanded in [12] where a dataset of audio-video recordings of single vehicles passing by the camera at a known constant speed was reported. The experimental results show that the proposed DL regression model was able to estimate the vehicles' speed with an average error of around 7 km/h. In [29], the authors proposed a vehicle detection and classification (VDC) system designed to track vehicle passages on bridges. The system exploits the unique acoustic signatures generated by vehicles as they interact with bridge expansion joints. A threshold-based algorithm was employed for vehicle detection, while a CNN model was trained for vehicle classification. In [32], audio classification is used to recognize emergency vehicle sirens from various background noises in order to act consequently on smart traffic management systems. The proposed ensemble model, incorporating different DL techniques, demonstrated an accuracy close to 100%. Vehicle identification from audio signals has also been applied to UAVs to address threats to civilian safety and national security [15], [36]. In [36], the authors recorded audio from flying drones in both indoor and outdoor environments, creating a dataset featuring ten different drone types. They then trained a CNN model, achieving a test set accuracy of over 95%. Specifically regarding TCI systems, we identified only two recent studies [23], [33]. The work

in [23] utilized mel classification for a different purpose than our framework since it focused on detecting and diagnosing faults in railway condition monitoring systems, specifically in railway point machines. The work in [33] proposed a system for the early detection of train arrival based on the recognition of train horn sounds from the traffic soundscape. To this aim, the authors developed a dataset with car horns, train horns, and background environmental noise and achieved 93% accuracy using a CNN.

To the best of our knowledge, the RATTLE framework represents the first audio-based TCI system to integrate multiple features—train passage identification, train classification, carriage counting, and speed estimation—into a single solution. Furthermore, in contrast to prior research on audio-based vehicular identification systems discussed in this section, our work addresses critical architectural challenges, including system scalability and DL optimization, to enable effective vehicle tracking across multiple locations.

Recent works confirm that mel-spectrogram features and compact CNNs are highly effective across domains (for example, cough recognition [38], vehicle speed estimation [12], siren detection [32], and UAV classification [15], [36]). However, we observe four recurring gaps when the goal is practical TCI at scale: (i) deployment constraints across the edge–cloud continuum are rarely discussed or evaluated; (ii) privacy and data governance considerations are often implicit, with vision-centric solutions [22], [35] carrying a higher privacy burden than audio-only sensing; (iii) links to SHM and multimodal pipelines are typically absent; and (iv) robustness to real-world conditions (cross-location generalization, fine-tuning effort, network limits) is seldom characterized beyond curated datasets. Our work addresses these gaps by: (a) explicitly analyzing processing placement across the continuum (Section III, Section VI); (b) adopting audio-only inference to reduce privacy exposure while maintaining accuracy (Section I, Section VI); (c) positioning TCI as a first-class input to SHM pipelines and outlining a multimodal path with vibration (Section VII); and (d) quantifying location-specific fine-tuning needs and edge feasibility with compact models (Section VI). Table 1 provides a per-paper comparison across focus, edge–cloud placement, privacy, modalities, SHM, and real-world constraints, and contrasts our framework accordingly.

III. RATTLE ARCHITECTURE

The RATTLE architecture comprises a data pipeline beginning from railroad monitoring and culminating in identifying several features of trains based solely on audio fingerprints. While envisioning our framework, and carefully planning where to locate each component of our architecture in the Edge-Cloud continuum, we reviewed and applied the best practices highlighted in [4] and [16]. Figure 1 depicts the bird's eye view of the RATTLE architecture, in which the processing flow is depicted at the top of the image, and the bottom of it shows the underlying infrastructure to deploy the architecture. The system comprises one or more IoT field systems that collect data from trains passing through monitored railroads, transmit them to the subsequent components of the RATTLE processing pipeline, and ultimately store the data on a cloud server. When deploying the system in a new environment (i.e., to monitor a new railroad), the system administrator can choose to adapt the global DL model for the specific characteristics of the new monitoring location. In this case, there is an *offline* phase before the system becomes operational, the operations of this phase are denoted in Fig. 1 by dotted-orange arrows.

In the offline phase, it is required to capture and label a few samples of audio and video recordings of trains from the new location to fine-tune the global model to this new monitoring location. Labeling can be done manually or using automated methods [11]. The number of samples needed to adapt the model depends on: (*i*) the diversity of characteristics at the new location compared to the original training dataset; (*ii*) the accuracy requirements of the scenario. Once the offline phase is completed, the model is deployed, and the system initiates performing real-time inference. Although optional, fine-tuning the models with even few samples can lead to substantial performance improvements as demonstrated by previous literature ([7]) and by our experiments in Sec. VI. We highlight that an IoT field system equipped with a camera is necessary to collect video data.

In *online* phase (denoted by the blue arrows in Figure 1), the identification of train features relies exclusively on audio signals. This choice offers several advantages, including (*i*) privacy enhancement, since video captures may introduce privacy concerns; (*ii*) cost reduction, when the IoT monitoring system must be deployed at multiple locations along the railway; (*iii*) light independence, since video capture relies on sufficient light for train classification, posing challenges in dark conditions; and (*iv*) easier deployment, as an IoT field system that records only audio does not require positioning to ensure a clear and steady view of the railway. Moreover, in Section VI, we demonstrate that we can achieve similar accuracy with audio classification as compared to state-of-the-art computer vision methods. The online and fine-tuning phases can be logically decoupled, allowing devices to execute them at different locations. For instance, the data collection to build the dataset can be deployed in one location, while the inference can be conducted in other, potentially multiple, spots.

The processing steps of RATTLE are decoupled from its underlying implementation – as we exemplify in Section VI by implementing different ML/DL algorithms for train classification. In detail, the stages of our proposed pipeline are:

1) **Data Collection**: audio and video (video recordings are required for offline phase only) data are recorded from trains passing through the railroad. A given noise threshold is set to trigger the start and stop of the data collection.
2) **Data Filtering**: heuristics remove invalid data points (e.g., recordings shorter than a threshold are deleted).

TABLE 1. Per-paper comparison across focus, edge–cloud placement, privacy, modalities, SHM, and real-world constraints. Descriptions are qualitative and based on each paper's stated scope.

Work	Focus	Edge–Cloud Placement	Privacy Concerns	Modalities	SHM	Real-world constraints
[22]	Vision-based wagon counting and identification	Not assessed	High (vision in public spaces)	Video	No	Evaluated on curated images; deployment aspects not discussed
[35]	Real-time wagon counting and number recognition (vision)	Not assessed	High (vision)	Video	No	Real-time focus; lacks cross-location/continuum characterization
[33]	Train arrival detection via horn	Not assessed	Low (audio)	Audio	No	Field-like data; no edge/network analysis
[23]	Fault detection of railway point machines by sound	Not assessed	Low (audio; equipment)	Audio	No (equipment monitoring)	Lab/field records; no cross-location fine-tuning
[5]	Vehicle speed estimation from roadside audio (mel features)	Not assessed	Low (audio)	Audio	No	Controlled passes; generalization across sites not assessed
[12]	Vehicle speed estimation from single acoustic sensor	Not assessed	Low (audio)	Audio	No	Controlled dataset; deployment constraints not analyzed
[32]	Emergency siren detection for traffic management	Not assessed	Low (audio)	Audio	No	High accuracy on dataset; no continuum/edge feasibility
[29]	Vehicle detection and classification under bridge expansion joint (voiceprint)	Not assessed	Low (audio at infrastructure)	Audio	No	Site-specific evaluation; no cross-location study
[36]	Large-scale UAV audio dataset and CNN classification	Not assessed	Low (audio)	Audio	No	Indoor/outdoor and distance variations; no edge analysis
[15]	UAV detection with GAN-augmented audio dataset	Not assessed	Low (audio)	Audio	No	Emphasis on augmentation; deployment concerns not covered
[3]	Vibration WIM for axle weight estimation (road traffic)	Edge (field deployment)	Low (vibration)	Vibration	Discussed (weight inference context)	Field-deployed; motivates multimodal audio+vibration
[38]	Cough recognition with mel-spectrograms and CNN	Not assessed	Low (audio; clinical)	Audio	No	Demonstrates mel+CNN effectiveness; not TCI-focused
[20]	Robust audio fingerprinting under noise/distortion	Not assessed	Low (audio)	Audio	No	Robustness analysis; not evaluated for TCI deployment
RATTLE (this work)	Audio-only TCI: class, length, speed	Edge+Cloud (analyzed; Secs. III,VI)	Low (audio-first)	Audio (vision used offline for fine-tuning)	Integrated (designed to feed SHM; multimodal path)	Cross-location fine-tuning; networking/placement trade-offs empirically characterized

3) **Feature Extraction**: relevant characteristics are extracted from the audio signal (e.g., frequency and amplitude) and the Mel Frequency Cepstral Coefficients (MFCCs) are generated and subsequently stored in a database. The mel-spectrogram is a graphic representation of the frequencies contained in an audio sample, and it is the standard data representation for audio-based DL [28]. Similarly, pertinent frames corresponding to the passage of trains are extracted from the video during the fine-tuning operations.
4) **Data Analysis**: supervised ML or DL algorithms are employed to classify the features extracted from the previous step and each model outputs one feature of the train.

Currently, RATTLE provides a DL model that captures the primary audio features of passing trains. For each specific task, the model is customized by incorporating an appropriate final linear layer tailored to that task. Further details of the model architecture are presented in Sec. V. The current TCI tasks supported by RATTLE are: (*i*) Train classification; (*ii*) Carriage counting; (*iii*) Train speed estimation.

When deploying the system, the user can choose which one of those models to deploy. Furthermore, due to RATTLE modular architecture, new models can be easily added to the system – that utilized or not the current common pre-trained model that is utilized for all TCI.

A. IoT FIELD SYSTEM

In this Subsection, we detail the IoT field system deployed to collect data on a single train track. We utilized the described IoT field system to collect a dataset, namely the *field* dataset, which is described in Sec IV. Although we proposed and designed an IoT field system for RATTLE, other systems with heterogeneous IoT components can be used for data collection, as demonstrated in the Sec. VI.

The RATTLE IoT field system was designed to fulfill the following requirements necessary for railroad monitoring:

- *Portability*: Our IoT system is designed for easy replication and deployment across diverse locations with minimal setup overhead and reliance on existing infrastructure. Such portability is crucial for expanding the training dataset (offline phase) and facilitating inference operations at multiple sites (online phase).
- *Edge-Friendly Processing*: Designed to operate in environments near the railway, our IoT system operates efficiently even in conditions with limited Internet connectivity. To minimize reliance on cloud resources, we explore utilizing lightweight DL models to execute inference tasks directly on edge, as detailed in Section VI.

The prototypal IoT system shown in Figure 2 comprises a sensing layer and an edge processing layer. The first includes

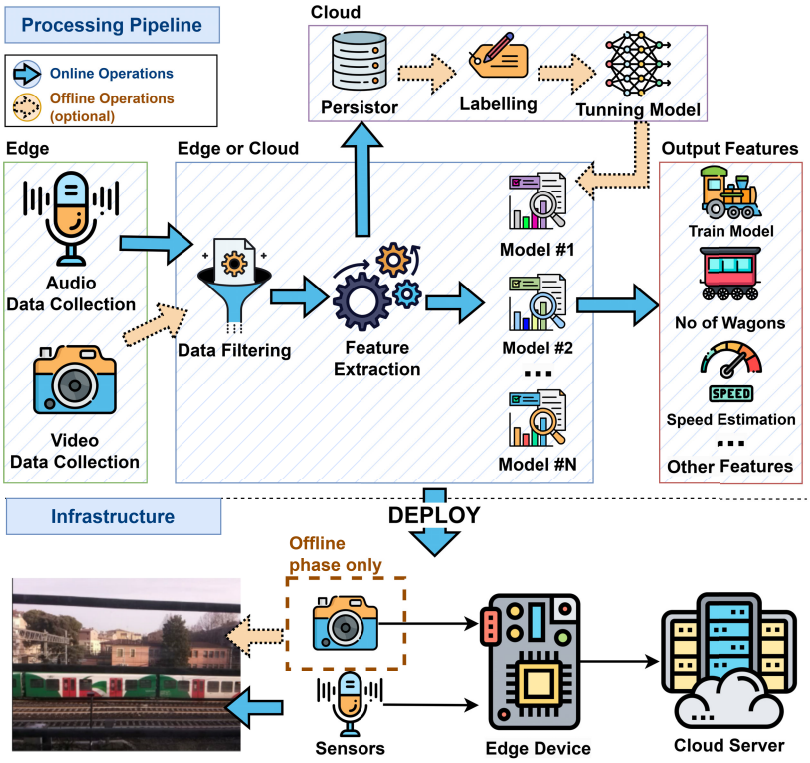


FIGURE 1. RATTLE Architecture depicting the processing pipeline and underlying computational infrastructure.

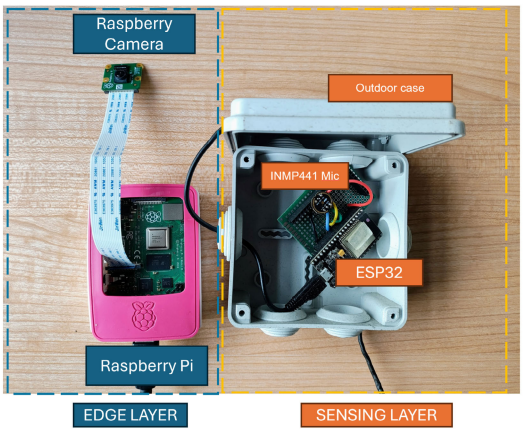


FIGURE 2. IoT System used for the data collection.

an INMP441 Omnidirectional Microphone connected to an ESP32 board, automatically triggering sound registration when detecting a new train passage. The second is composed of a Raspberry Pi 3 Model B with a connected camera for video recordings and acts mainly as a temporary data storage, filter, and feature extractor. The microphone is always listening and monitoring the amplitude of the received signal. Since the locomotive's arrival could be meaningful for train recognition, to avoid losing any information, the device always keeps a cache of fixed length storing the last N seconds of audio recording—N is a user-set parameter, and its value depends on the deployment location of the IoT field system. Whenever the device detects a sound with an amplitude greater than a predefined threshold ψ, it triggers a new registration, sending the recorded sound waves to the edge node through Web sockets. Likewise, also the edge node starts recording with the camera if it is in the offline phase. The recording lasts until the amplitude is below the ψ threshold, and we add padding of N seconds to ensure we capture the whole train. In our experiment, we set ψ to 103 dB; the value has been found experimentally as the optimal balance between the recording length and occurrences of false positives. However, the threshold is location-dependent and should be properly tuned for each setup.

B. EDGE-CLOUD CONTINUUM SUPPORT

Different deployment scenarios often entail non-compatible infrastructure, posing challenges regarding adaptability and the software components distribution across the edge-cloud continuum [16]. To adhere to the requirements of adaptability and components distribution, the RATTLE components are not bound to a given processing placement, as Figure 1 denotes; indeed, the data filtering and feature extraction steps can be performed at either the edge or cloud locations depending on the scenario constraints. For instance, in *network-constrained* scenarios, the data filtering, feature extraction, and even the classification stages of Figure 1 can be placed at the edge to reduce the data volume transferred from edge

to cloud. On the other hand, this placement configuration is unfeasible in *resource-constrained* scenarios, in which the edge nodes cannot perform computation tasks other than data collection. In Subsection VI-E, we characterize the RATTLE performance in diverse placement configurations across the edge-cloud continuum.

IV. DATASET

In this Section, we explore the two datasets created for training the models, as explained in Section V. We distinguish between the *global* dataset and the *field* dataset, where each sample consists of video and audio recordings of a passing train. The *global* dataset aggregates samples from multiple locations, including a wide variety of classes and recording conditions. These samples are extracted from videos recorded by Akiem483.[1] This provides a diverse range of scenarios, making it a valuable resource for improving the generalization capabilities of the model. The *field* dataset consists of samples collected from a specific location by the IoT field system described in Sec. III and is used for fine-tuning the model to the unique conditions of that station. *Field* datasets are more constrained, characterized by fewer classes and a narrower distribution of recording conditions, as they typically involve consistent hardware setups, fixed camera angles, and standardized microphone distances. In our setup, the classes represented in the *field* dataset differ from those in the *global* dataset for two key reasons: (1) we utilized the dataset collected in [10] without additional classification efforts, and (2) this simulates a realistic scenario where a field node is focused on monitoring specific classes that may differ from the broader set of classes in the global dataset. This distinction reflects practical deployment conditions, where localized monitoring requirements often diverge from the comprehensive scope of a global dataset. For reproducibility we made the *field* dataset [9] available.

As a preliminary step, we removed corrupted audio or video samples from both datasets. Then, we segmented the audio into overlapping chunks of equal length, determined by user-defined window and step sizes. The total number of chunks for an audio signal can be calculated using Equation 1, where l is the sample length of the audio signal, b is the bit-rate, w is the window size, and s is the step size. This segmentation allowed us to maintain the temporal dimension while computing features.

$$c = 1 + \frac{l - wb}{sb} \tag{1}$$

As shown in Algorithm 1, the process of feature extraction consists of three main stages. First, the raw audio signal is loaded and divided into overlapping chunks of length w with a step size of s (l. 1). Each chunk is processed to extract its short-term features (ll. 3–9). For each extracted feature, its delta values—representing the rate of change—are also computed to capture temporal dynamics (ll. 10–14). The resulting features are then summarized into the mid-term descriptors by computing their mean and standard deviation over the entire signal (ll. 17–18). The extraction process generates 136 mid-term features from raw audio signals, computed from 34 short-term features – including MFCCs spectral centroid, and other spectral and temporal descriptors – and their delta values as detailed in Table 2. The final feature set follows the naming convention:

$$[delta]feature_name[_mean|_std] \tag{2}$$

TABLE 2. Short-term features extracted.

Feature	Name	Description
1	Zero Crossing Rate	Frequency of signal sign-changes in a frame.
2	Energy	Normalized sum of signal squares in a frame.
3	Entropy of Energy	Entropy of normalized sub-frame energies.
4	Spectral Centroid	Spectrum's center of gravity.
5	Spectral Spread	Spectrum's second central moment.
6	Spectral Entropy	Entropy of normalized spectral sub-frame energies.
7	Spectral Flux	Squared difference between successive spectral magnitudes.
8	Spectral Rolloff	Frequency below 90% of spectral energy.
9–21	MFCCs	Cepstral representation using mel scale bands.
22–33	Chroma Vector	Spectral energy over 12 pitch classes.
34	Chroma Deviation	Standard deviation of chroma coefficients.

As shown in Figure 3, the MFCCs features are the most strongly correlated. Although our analysis identified MFCCs as the audio features with the strongest correlation, we opted to use mel-spectrograms for the rest of analyses. This decision is justified by the inherent relationship between the two: MFCCs are derived from mel-spectrograms through a discrete cosine transform, and the mel-spectrogram serves as their foundational representation. By using mel-spectrograms directly, we are able to preserve the full time-frequency representation, which is particularly advantageous when working with DL models like CNNs. Additionally, unlike MFCCs which reduce dimensionality and decorrelate features for simpler models, mel-spectrograms provide richer and more interpretable information about the audio signals. Finally, from each audio sample, we extracted the corresponding mel-spectrogram. An example of a Frecciarossa train mel-spectrogram is depicted in Figure 4.

Due to differences in acquisition techniques, separate data filtering and labeling were applied to each dataset, as described in Sections IV-A and IV-B.

A. GLOBAL DATASET LABELING

For the *global* dataset, we collected a total of 3,791 samples. The labels for these samples were automatically assigned

[1] https://www.instagram.com/akiem_e483/

Algorithm 1 Audio Feature Extraction

Require: Raw audio signal A, window size w, step size s
Ensure: Set of 136 mid-term features F

$chunks \leftarrow \text{SegmentAudio}(A, w, s)$ *# Segment into overlapping windows*
$F_{short} \leftarrow \{\}$
$features \leftarrow \{\text{mfcc}, \text{spectral_centroid}, \ldots\}$ *# 34 base features to extract*
for each *chunk* in *chunks* **do**
  $base_features \leftarrow \{\}$
  for each f in *features* **do**
    $value \leftarrow \text{ExtractFeature}(chunk, f)$
    $base_features \leftarrow base_features \cup \{value\}$
  end for
  $delta_features \leftarrow \{\}$
  for each bf in *base_features* **do**
    $value \leftarrow \text{ComputeDeltaValues}(bf)$
    $delta_features \leftarrow delta_features \cup \{value\}$
  end for
  $F_{short} \leftarrow F_{short} \cup \{base_features, delta_features\}$
end for
$F_{mean} \leftarrow \text{Mean}(F_{short})$ *# Compute 68 mean features*
$F_{std} \leftarrow \text{StandardDeviation}(F_{short})$ *# Compute 68 std features*
$F \leftarrow F_{mean} \cup F_{std}$
return F

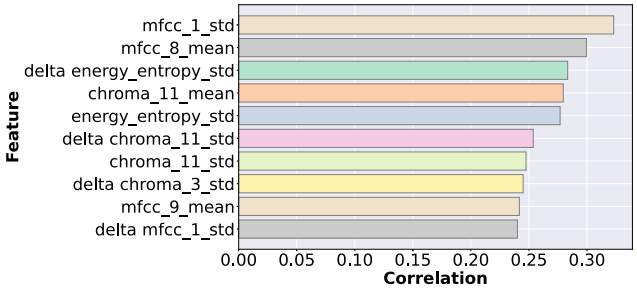


FIGURE 3. Most strongly correlated features on the audio dataset.

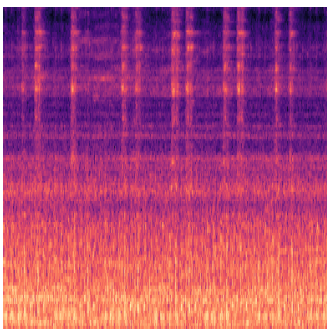


FIGURE 4. Mel-spectrogram of a Frecciarossa train.

by parsing the corresponding video titles with regular expressions, which extracted the service category, group, and series number. Samples belonging to freight service categories were excluded due to the wide variability in train

FIGURE 5. Example of trains grouped in the ETR4XX category.

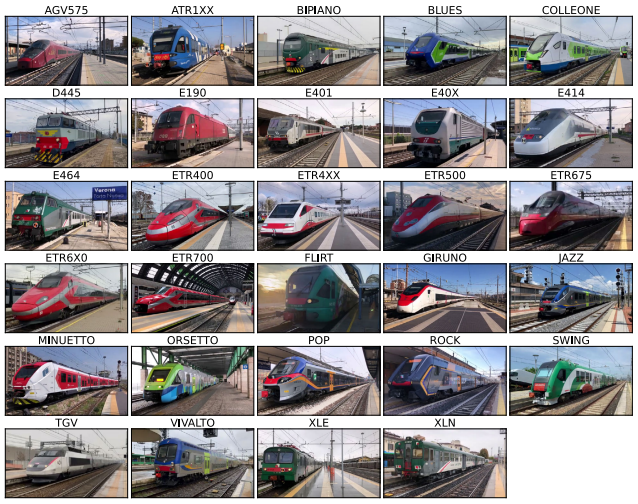


FIGURE 6. Examples of images for each train category in the global dataset.

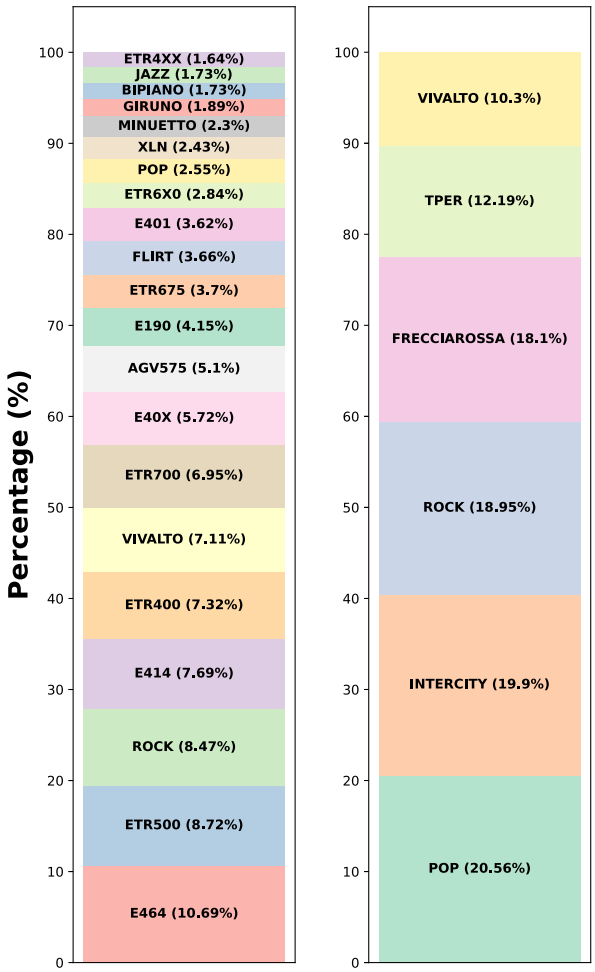
groups and compositions, which made standardization and grouping impractical.

The remaining samples were grouped based on the main train groups identified in the dataset. While this process was largely automated, certain groups required manual classification. For example, the *E464* group, which includes both standard regional trains and *Vivalto* double-deck trains, was classified on a per-sample basis. Similar train groups were aggregated into macro-classes, as depicted in Figure 5. A detailed breakdown of the aggregation process is provided in Appendix. The final clustered dataset includes a total of 29 classes, with examples shown in Figure 6.

After completing the aggregation process, the dataset was further refined by removing eight under-represented classes that lacked sufficient samples for effective training. The final global dataset labels' distribution is depicted in Figure 7(a)

B. FIELD DATASET LABELING

For our *field* dataset, we collected audio samples and video recordings of each train passage over a period exceeding four months, reaching a total of approximately 19,633 samples. To expedite the manual classification process, we extracted one frame per second from the video footage associated with the filtered audio files. These frames facilitated the annotation process by providing visual references for train identification. Given the extensiveness of the dataset, not all samples could be manually classified. Those that were successfully annotated were categorized into seven train classes, as shown in Figure 8. The *Frecciarossa* and *Intercity* categories encompassed multiple train models within the



(a) Global (b) Field

FIGURE 7. Labels' distribution within the global (a) and field (b) datasets.

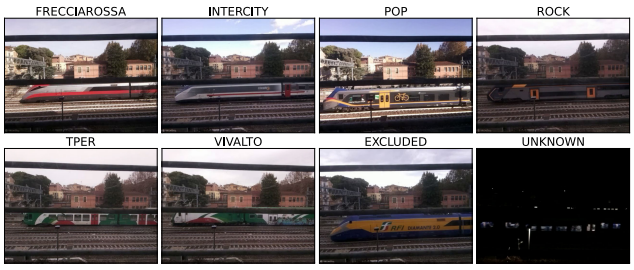


FIGURE 8. Examples of images for each train category in the field dataset.

same service type. Samples that could not be confidently assigned to a specific class, either due to insufficient visual information or ambiguous train characteristics, were grouped under the *unknown* label. Additionally, an *excluded* label was assigned to unclassifiable instances, such as rare train models or cases where multiple trains were passing simultaneously. The manual classification process yielded a ground truth dataset containing 2.116 classifiable samples. Figure 7(b) provides an overview of the label distribution obtained from the dataset.

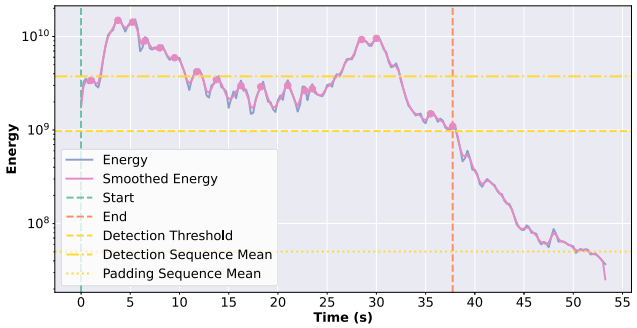


FIGURE 9. Energy profile and transit detection of a train passage.

In addition to training class and carriage count, the field dataset was annotated with the approximate speed of passing trains by estimating their estimated length and transit time. The average carriage length for each train class was multiplied by the number of carriages to estimate train length. The transit time of each train was determined by analyzing the energy profile of its corresponding audio sample, as detailed in Algorithm 2 and illustrated in Figure 9. First, the audio signal was segmented into two parts: the final p seconds of recording – 5 in our setup – assumed to be background noise (l. 2), and the remaining portion containing the train's passage (l. 1). The energy of both segments was computed using a sliding window approach (ll. 5–6) and averaged to obtain representative energy levels (ll. 7–8). A dynamic detection threshold was then derived by interpolating between these two energy levels (l. 9). Next, the algorithm processed the full audio signal to compute its energy profile (l. 10), which was then smoothed to reduce fluctuations (l. 11). The first frame exceeding the detection threshold was identified as the estimated start of the train's passage (ll. 13–18). Similarly, peaks in the energy profile were analyzed in reverse order to locate the last frame above the threshold, marking the estimated end of the passage (ll. 19–26). The time difference between these events defined the transit duration, which was then used to approximate the average train speed. While not perfectly precise, these estimates provided a solid foundation for the experiments discussed in Section VI-D3.

V. MODEL ARCHITECTURE

This section outlines the architecture of the custom DL model designed for the task of TCI. The discussion is divided into three parts: the model's architecture, its deployment configurations, and its memory requirements.

As introduced in Section I, the model is designed to perform three distinct tasks: train classification, carriage counting, and speed estimation. These tasks fall into two categories – classification (for train types) and regression (for the number of carriages and speed). They are executed using audio features as input, with a focus on maintaining a compact memory footprint. The proposed model, referred to as CustomNet, is a CNN specifically optimized for processing mel-spectrogram inputs of dimensions $64 \times 64 \times 3$. The architecture is designed to hierarchically extract features

Algorithm 2 Estimating Train Transit Time From Audio

Require:
Raw audio signal A,
Sampling rate sr,
Threshold factor τ,
Padding length p,
Ensure: Estimated transit time $(T_{\text{start}}, T_{\text{end}})$

$S \leftarrow A[0 : \text{len}(A) - p \cdot sr]$ *# Unpadded audio segment*
$P \leftarrow A[\text{len}(A) - p \cdot sr : \text{len}(A)]$ *# Padding audio segment*
$w \leftarrow sr/2$ *# 500 ms window size*
$s \leftarrow w/2$ *# 50% overlap*
$E_S \leftarrow \text{ComputeEnergy}(S, w, s)$
$E_P \leftarrow \text{ComputeEnergy}(P, w, s)$
$E_{S,\text{mean}} \leftarrow \text{Mean}(E_S)$
$E_{P,\text{mean}} \leftarrow \text{Mean}(E_P)$
$th \leftarrow (1 - \tau) \cdot E_{P,\text{mean}} + \tau \cdot E_{S,\text{mean}}$ *# Detection threshold*
$E \leftarrow \text{ComputeEnergy}(A, w, s)$
$E_{\text{smooth}} \leftarrow \text{MovingAverage}(E, 3)$ *# Smoothing with 3-frame window*
$T_{\text{start}} \leftarrow 0, \quad T_{\text{end}} \leftarrow 0$
for $i = 0$ to $\text{len}(E_{\text{smooth}})$ **do**
  if $E_{\text{smooth}}[i] > th$ **then**
    $T_{\text{start}} \leftarrow i \cdot s/sr$
    break
  end if
end for
$peaks \leftarrow \text{FindPeaks}(E_{\text{smooth}})$
for $j = \text{len}(peaks)$ down to 0 **do**
  $i \leftarrow \text{peaks}[j]$
  if $E_{\text{smooth}}[i] > th$ **then**
    $T_{\text{end}} \leftarrow i \cdot s/sr$
    break
  end if
end for
return $(T_{\text{start}}, T_{\text{end}})$

from the input spectrograms through a series of convolutional layers, followed by a fully connected network that generates task-specific predictions.

The convolutional block consists of four sequential convolutional layers, each followed by instance normalization and ReLU activation. The convolutional layers employ progressively increasing the number of feature maps (32, 64, 128, and 256) to capture features at multiple scales, while the use of instance normalization ensures stable training by normalizing activations within each sample. Strides of 2 are applied in each convolutional layer to reduce spatial dimensions, enabling the model to focus on increasingly abstract features. The dimensions of the feature map after each convolutional layer are computed using Equation 3:

$$D_{\text{out}} = \left\lfloor \frac{D_{\text{in}} + 2P - K}{S} + 1 \right\rfloor \tag{3}$$

where D_{in} represents the input dimension (height or width), P denotes the padding, K is the kernel size, and S corresponds to the stride. Equation 3 ensures that the spatial dimensions of the feature maps are systematically reduced, allowing the model to hierarchically extract and condense information from the input spectrogram.

Following the convolutional block, an adaptive average pooling layer is used to reduce the spatial dimensions to a fixed size (1×1), ensuring that the model can handle inputs of varying sizes while preserving the channel-wise information. The pooled features are then flattened and passed through a fully connected (FC) block, which consists of two linear layers separated by a LeakyReLU activation, batch normalization, and dropout. The dropout layer mitigates overfitting, while batch normalization stabilizes training. The final linear layer produces an output whose dimensionality depends on the task: a single neuron for regression or N neurons for classification, where N corresponds to the number of target classes. A visual representation of the model architecture is shown in Figure 10.

In the following subsections, we explore potential deployment configurations and the requirements of our model. Specifically, Subsection V-A details the deployment configurations evaluated in our experiments, while Subsection V-B analyses the memory requirements for storing the model and its performance during inference.

A. PROPOSED DEPLOYMENT CONFIGURATIONS

When deploying a new field system, a copy of the global model is instantiated on the new field node. At this stage, the system administrator can decide whether to fine-tune the model before bringing the system online, including selecting which layers to fine-tune and which to keep frozen. We propose three distinct configurations for administrators to choose from, each varying in the required system downtime and the number of field samples needed for setup:

- **Configuration 1** : The model is used without any modifications or fine-tuning. It also ensures zero downtime during deployment. This configuration is represented in Figure 11(a).
- **Configuration 2** : All convolutional layers are frozen, and only the FC layers are updated during training. This approach focuses on adapting the features extracted by the pre-trained convolutional layers to the specific task. A visual representation of this configuration is provided in Figure 11(b). This setup requires an offline period to gather sufficient samples for updating the model.
- **Configuration 3** : This configuration extends Configuration 2 by including the last convolutional layer in addition to the FC layers. This allows the model to adapt both its high-level feature extraction and classification components to better fit the target dataset. A visual representation of this configuration is shown in Figure 11(c).

In general, fine-tuning has been proved to greatly improve task performance and allow transfer-learning with minimal

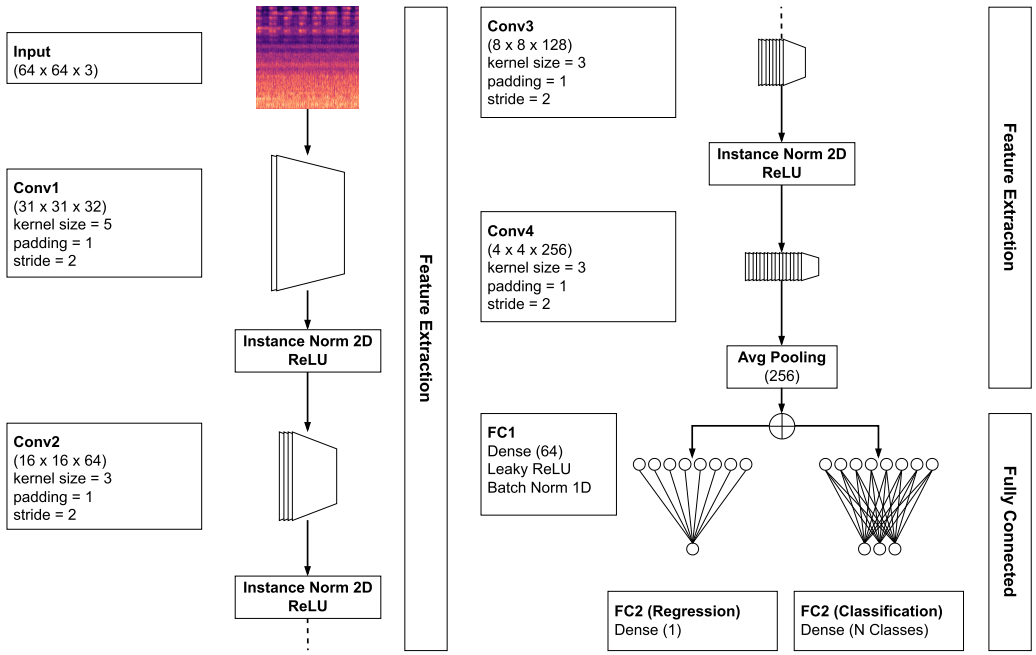
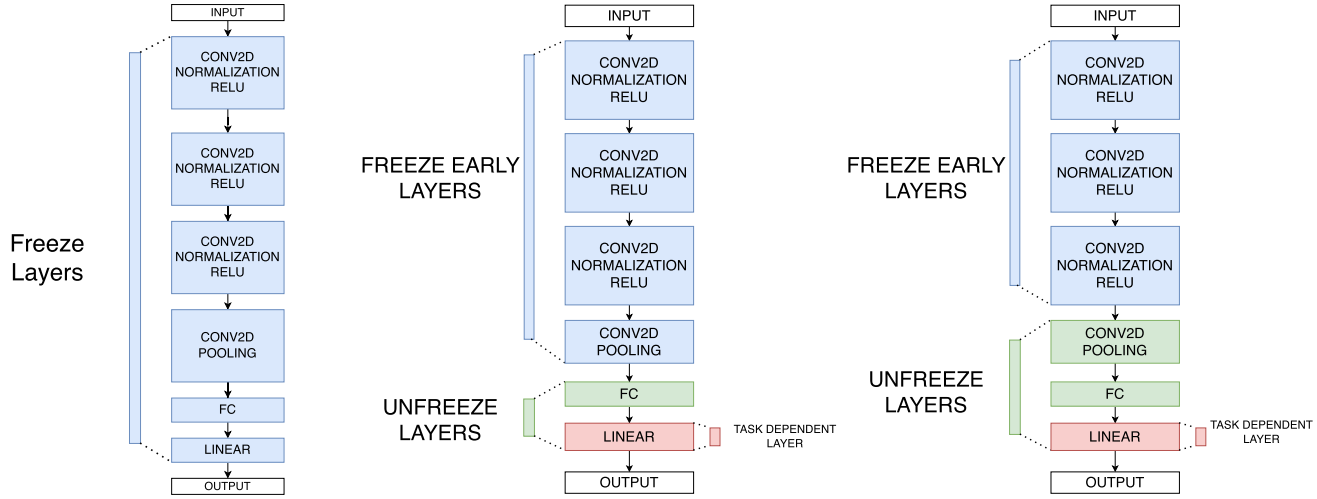


FIGURE 10. Model architecture depicting feature extraction layers and task dependent fully connected layers.



(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

FIGURE 11. (a) No fine-tuning: all layers are frozen. (b) Lighter fine-tuning: all convolutional layers are frozen, and only the fully connected layers are retrained. (c) Deeper fine-tuning: the last convolutional layer is unfrozen and trained alongside the fully connected layers.

data collection and retraining required [7], making Configurations 2 and 3 our recommended choices for model deployment. In Section VI, we evaluate the proposed deployment configurations and present our findings on their performance.

B. MEMORY REQUIREMENTS

The memory requirements of the proposed model are determined by the number of parameters and the activation maps generated during inference. The total number of parameters for each layer is calculated as follows:

- **Convolutional Layer**:

$$\text{Parameters} = (K_h \times K_w \times C_{\text{in}} + 1) \times C_{\text{out}} \quad (4)$$

where $K_h \times K_w$ is the kernel size (height $\times$ width), C_{in} is the number of input channels, C_{out} is the number of output channels, and the $+1$ accounts for the bias term.

- **Fully Connected Layer**:

$$\text{Parameters} = (S_{\text{in}} + 1) \times S_{\text{out}} \quad (5)$$

where S_{in} is the input size and S_{out} is the output size. The $+1$ accounts for the bias parameters.

- **Normalization Layers**:
  Instance normalization layers do not contribute parameters, while batch normalization layers introduce 2 × Input Size parameters (gamma and beta).

Table 3 summarizes the number of parameters and memory occupation for each layer of the model, distinguishing between regression (output size = 1) and classification (output size = 6) tasks.

Peak Memory Requirement: During inference, activation maps are managed dynamically. Only one activation map is stored at any given time, minimizing peak memory usage. The peak memory requirement at inference time is calculated as the sum of the largest activation map and the model's parameters. The largest activation map, generated by the first convolutional layer, is computed as

$$\text{Activation Size} = 31 \times 31 \times 32 \times 4 \text{ bytes} = 123,008 \text{ bytes} \approx 123 \text{ KB}.$$

where:

- 31×31 is the spatial size of the output feature map,
- 32 is the number of output channels,
- 4 bytes is the size of a 32-bit float.

The model parameters require **1.63 MB**. Taking into account the largest activation map, the total peak memory requirement for both regression and classification tasks is approximately **1.75 MB**.

TABLE 3. Number of parameters and memory occupation per layer.

Layer	Params (Reg)	Mem (Reg) (Bytes)	Params (Cls)	Mem (Cls) (Bytes)
Conv1	2,432	9,728	2,432	9,728
Conv2	18,496	73,984	18,496	73,984
Conv3	73,856	295,424	73,856	295,424
Conv4	295,168	1,180,672	295,168	1,180,672
BatchNorm1d	128	512	128	512
FC1	16,448	65,792	16,448	65,792
FC2	65	260	390	1,560
Total	406,593	1,626,372	406,918	1,627,672

We successfully conducted our edge deployment test on a Raspberry Pi 3 Model B, whose hardware capabilities vastly exceed our model's requirements. Indeed, the compact dimension of our network and the modest memory requirements at inference time make CustomNet suitable to be deployed on microcontroller boards with very constrained hardware capabilities. To give an example, the ESP32-S3 microcontroller supports up to 32 MB of external Pseudostatic RAM, which can be mapped into the memory address space and used for storing activations and model parameters.

VI. PERFORMANCE EVALUATION

In this section, we present the performance evaluation of the system described in the previous sections. We describe the methodology in details including procedures and techniques employed. We evaluate the performance of the proposed models with three experimental goals including: (*i*) train classification, (*ii*) counting the number of carriages and (*iii*) train speed estimation.

For the classification task, we evaluated model performance using accuracy and F1 score. These metrics were chosen because they provide a balanced assessment of classification performance, without imposing specific penalties on false positives or false negatives. Since our classification task does not involve critical constraints—such as those found in medical diagnosis, where false negatives are particularly costly—accuracy and F1 score offer an effective way to gauge overall model reliability. For regression tasks, we used Mean Absolute Error (MAE). MAE was selected because it directly quantifies the average prediction error in the same measurement units as the target variables. This makes it particularly useful for interpreting the model's performance in practical terms, providing a clear understanding of how far predictions deviate from actual values.

To provide a clear understanding of the experimental evaluation process, the following sections are structured as follows. Subsection VI-A explains the data preprocessing methodology in detail, and discuss the model's configuration and preprocessing steps implemented before evaluation. Subsection VI-B compares the performance and model size of our proposed network with state-of-the-art ML and DL models. Finally, the results of experiments involving both the global model (VI-C) and the field model (VI-D) are presented.

A. EXPERIMENTAL SETUP

In this section, we present a brief summary of the preprocessing steps that were applied prior to the analysis. First, the images were resized to a fixed dimension of 64×64 to ensure a uniform input dimension. Then, they were normalized by applying to each image a mean of 0.5 and a standard deviation of 0.5 to standardize the pixel values to be in the range $[-1, 1]$. This speeds up convergence during training, reduces the risk of vanishing or exploding gradients, and finally improves numerical stability. The dataset was then divided into training (60%), validation (20%) and testing (20%) subsets ensuring robust model evaluation avoiding overfitting problems. A final step was to incorporate class weights into the cross-entropy loss function to address the imbalance of the remaining classes. This ensures that the under-represented classes contribute proportionally to the loss, preventing the model from being biased towards the more common classes. The weights w were calculated using the Equation 6.

$$\forall\, c \in \mathit{Classes}, \quad w_c = \frac{n_{\text{samples}_c}}{n_{\text{classes}} \cdot n_{\text{samples}_c}} \tag{6}$$

To ensure optimal performance, hyperparameter tuning was conducted for all models discussed in Section VI-B, wherever applicable. This process aimed to refine both architectural and training parameters to achieve a balance between accuracy and generalization. However, since our primary focus is on presenting our proposed CustomNet model, Table 4 provides only its final configuration, as result

of an hyperparameter optimization, including the selected hyperparameters. The complete architecture of CustomNet is detailed in Section V.

TABLE 4. Hyperparameter configuration.

Hyperparameter	Chosen Value
Number of Convolutional Layers	4
Pooling Function	Adaptive Average
Learning Rate	1×10^{-2}
Patience	10
Hidden Size	256
Dropout Rate	0.2
Loss	Cross-Entropy
Optimizer	AdamW

B. MODEL VALIDATION

In this section, we aim to validate our proposed CNN against state-of-the-art models on the classification task. To this end, we investigate the performance of CustomNet against various state of the art ML and DL models, and a Feed Forward Neural Network (FFNN), trained on the field dataset as shown in Figure 12. The data have been preprocessed according to the procedure outlined in Section VI-A. During this experiment, we tested our models using multiple features, as detailed in Section IV, and compared the resulting classification accuracies, F1 scores, and storage requirements. Overall, the deep models excelled at the task at hand, all exceeding 95% accuracy and F1 score. Most notably, our proposed CustomNet matched in performance Vision Transformer (ViT), the best performing state-of-the-art vision model, at a fraction of the storage requirement (ViT$\approx$330 MB and CustomNet$<$2 MB). Indeed, both models exceeded the 98% threshold in the observed metrics, but the extremely compact network size of CustomNet makes it especially suitable for the Cloud-Edge continuum. Furthermore, ViT and MobileNetV2 (MN2) reached their highest performance while trained on video frames, which are dependent on the visibility conditions of at the time of the train's passage. On the other hand, CustomNet achieved 98.5% accuracy and F1 score using mel-spectrograms as input features, making it also suitable for low-visibility sensing.

C. GLOBAL MODEL PERFORMANCE

In this section, we analyze the performance of the global model, which is hosted in the cloud system and trained on the global dataset. This dataset encompasses a significantly larger number of train classes compared to the field datasets, enabling the model to learn a more comprehensive representation of the task. The primary utility of the global model lies in its ability to extract significant features for TCI. To evaluate its effectiveness, we assess the classification performance of the global model, which serves as a proxy for its capability to extract distinguishing features from the input data.

Figure 13 shows the confusion matrix derived from the training of the described model. As we can see from the figure, the model achieves good results in the train classification, despite the large number of classes originating from different sources and domains. Specifically, the model achieves an accuracy of 70.2% and an F1 score of 70.4%, demonstrating its effectiveness in handling this challenging classification task.

D. FIELD MODEL PERFORMANCE

Following the evaluation of the global model on the cloud, the final weights were stored and made available to the field systems. These weights enable field systems to employ fine-tuning based on their tasks. As mentioned above, in our experiments we evaluated the model for three different tasks. In each of them, the same architecture of the global model was used, with the only exception of the final linear layer, whose output is replaced depending on the different task or number of classes required by the system. With regard to the classification task, the model was employed in all three configurations described in V. With regard to the regression task, only configurations 2 and 3 were employed, as it was not feasible to directly apply a model trained for classification to this specific regression problem. In addition, in order to maintain consistency among experiments and allow for comparative analysis, the same preprocessing steps described previously were applied. Finally, the field dataset, as outlined in Section IV, was utilized for each experiment. This dataset represents a subset of the global dataset collected by the field system. However, in certain cases, such as the experiments discussed below, it may include a distinct set of classes. These classes could either form a superset of the train categories in the global model or represent an entirely different set. Therefore, during the experiments involving configuration 1, it was necessary to map the classes predicted by the global model to their corresponding categories in the field dataset. The following is organized into three subsections. In VI-D1 we will explore the performance on the classification task. In VI-D2 and VI-D3 we focus on evaluating the model on the regression task.

1) TRAIN CLASSIFICATION

The results of the configuration 1 are presented. The focus of this analysis is on evaluating the model's capacity to classify train classes in the absence of any fine-tuning. This enables an assessment of the robustness of the model when applied to data from diverse and challenging conditions. Figure 14 shows the confusion matrix detailing the performance for each class. The model achieved an accuracy of 53% and an F1 score of 0.61%, demonstrating its ability to classify train classes despite a certain level of error.

To clarify, the class *missing* includes all train categories from the global dataset (IV) that are not present in the field dataset, which explain the absence of predictions for the mentioned class. Conversely, the classes *Frecciarossa*, *intercity* and *TPER* represent supersets of groups of classes present in the global dataset. Table 5 presents the mapping of local labels to their corresponding classes. Any label

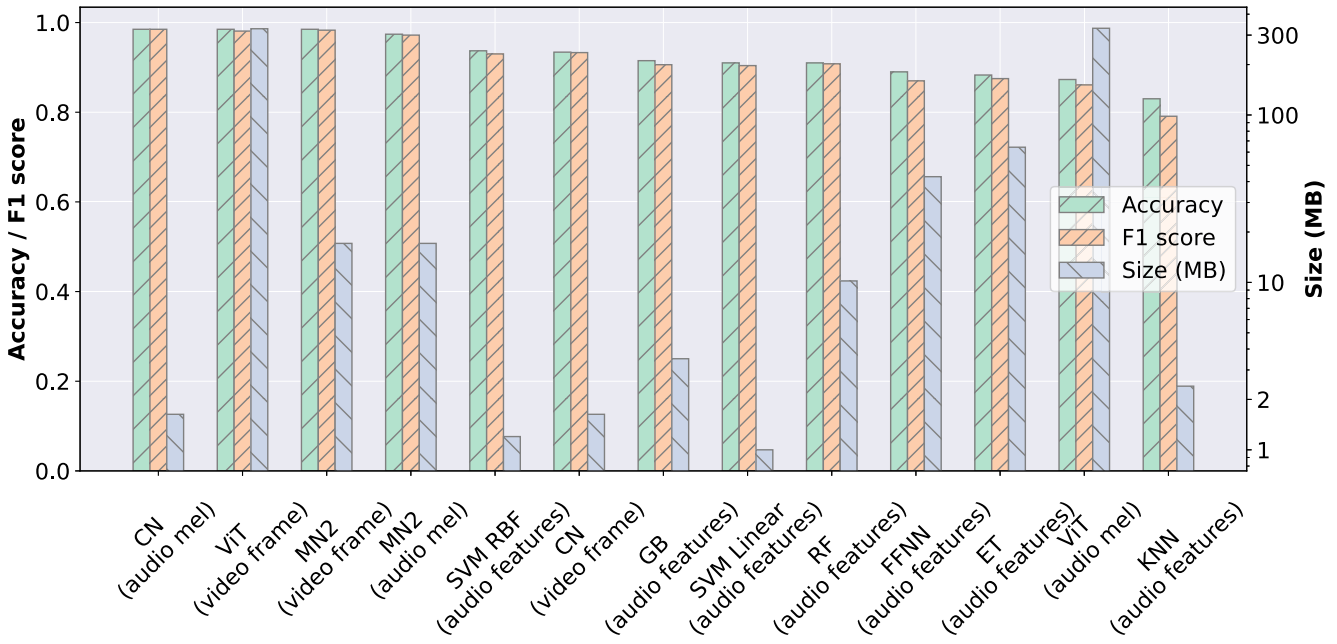


FIGURE 12. Comparison of classification accuracy, F1 score, and model size, ordered by descending accuracy, across tested models trained on local dataset, with annotated input type.

TABLE 5. Class mapping.

Global Dataset Class	Field Dataset Class
E401	Intercity
E414	Intercity
ETR400	Frecciarossa
ETR500	Frecciarossa
ETR6X0	Frecciarossa
ETR700	Frecciarossa
FLIRT	TPER

not explicitly listed in the table is categorized under the *"missing"* class.

The initial configuration of the model revealed the need for performance improvements. To address this, subsequent experiments were conducted with configurations 2 and 3, focusing on fine-tuning the model. Figure 15 and Figure 16 show the performance comparison in terms of accuracy and F1 score between the models fine-tuned as described above.

As observed in the figures, with a dataset comprising 540 training samples, the model achieved a 96% accuracy and F1 score through fine-tuning, signifying an enhancement over the 53% accuracy obtained with the global model's inference. Moreover, the outcomes reveal that the unfreezing of additional layers results in better performance. Finally, it is important to note that fine-tuning exclusively the linear layers resulted in impressive performance, achieving 95% accuracy when all the data samples were used. The Figure 17(a) and Figure 17(b) confirm the results by detailing the models' performance on the single classes.

We examined the correlation between sample length and classification accuracy to determine the optimal sensing length. To achieve this, we grouped the local ground truth samples into subsets based on their length. Each subset included samples in a four-second window, starting from the minimum sample duration in the dataset. Subsets with fewer samples were then grouped to ensure the reliability of the results. We finally computed accuracy and F1 Score metrics across all subsets. As shown in Figure 18(a) and in Figure 18(b), classification accuracy remains consistent as the sample length increases from 10 to 30 seconds, with performance converging in terms of both accuracy and F1 score. This suggests that starting with a sample length of 10 seconds is sufficient to achieve optimal results. In addition, these results are consistent with previous observations, confirming that deeper fine-tuning produces better performance than lighter fine-tuning, demonstrating its effectiveness in achieving higher accuracy and F1 Score.

Finally, we investigated the impact of the locomotive audio fingerprint on the classification task. To this end, we computed the accuracy and F1-score of our model using only the central portion of each audio sample. To extract the middle segment of each audio sample, we truncated the initial and final 25% of the interval where the train transit was detected. Resulting samples shorter than 3 seconds where excluded. Results of this experiment, shown in Figure 19(a) and Figure 19(b), suggest that we are able to correctly classify the train without relying on the sound of the first and last carriage, which is crucial for classifying different convoys sharing the same locomotive. Again, the comparison between the fine-tuning levels reveals analogous results to those

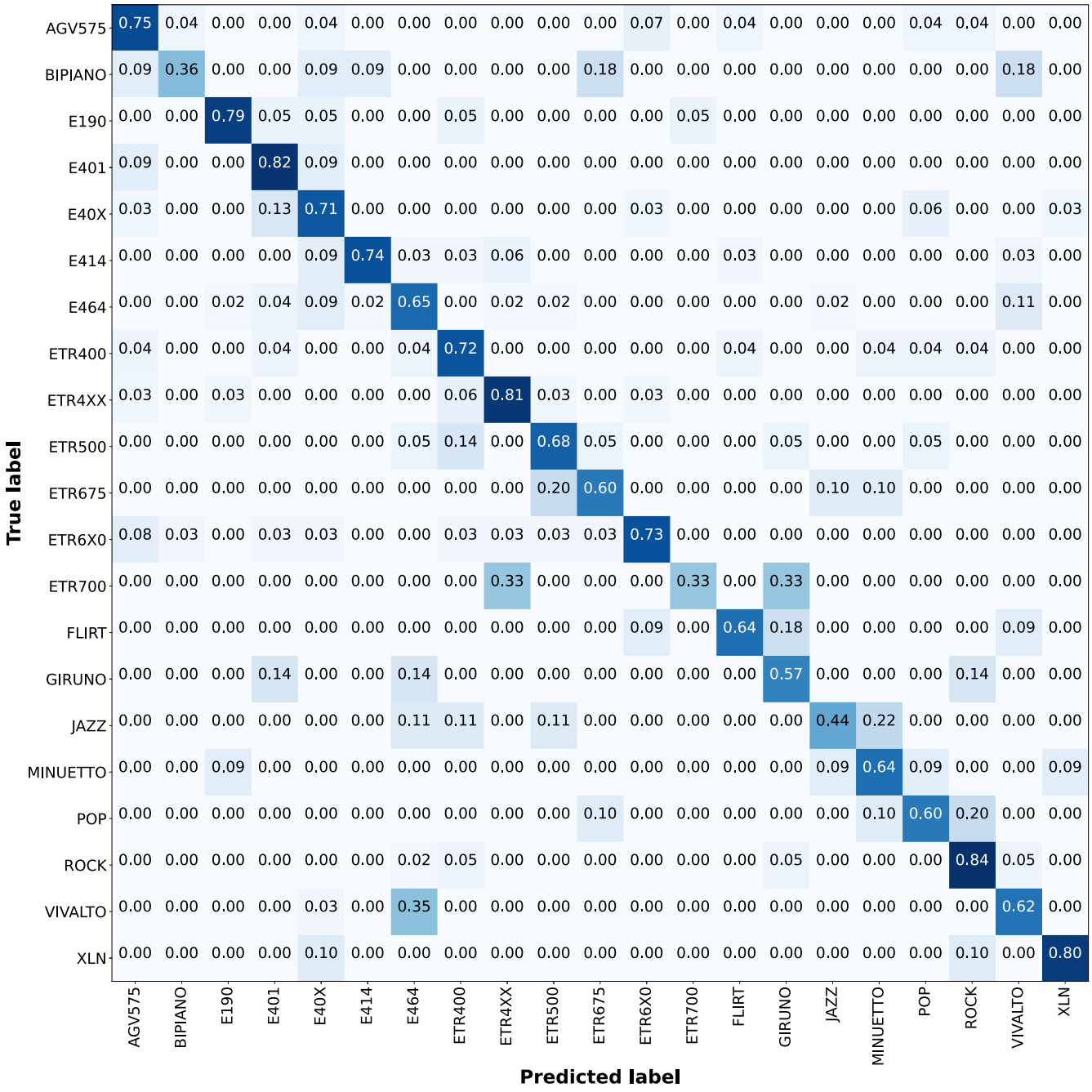


FIGURE 13. Global Model Confusion Matrix.

previously described, with enhanced performance observed at deeper levels of fine-tuning. Furthermore, our results show a slight improvement over the performance of shown in 19, this small variation could be the result of cleaner audio in the central portion of the audio sample.

2) CARRIAGE COUNTING

After evaluating the model's performance on the classification task, we shifted our focus to assessing its ability to predict the number of train carriages. The same dataset used for the classification task was employed for this regression task. Architecturally, we modified the model's final layer to adapt it for regression. Regarding parameters, we initialized the local model using the pre-trained global model parameters, as previously described. As demonstrated in Figure 20, the model's performance in terms of MAE is illustrated. On initial observation, the plot presents a pattern comparable to the classification results. With a reduced dataset comprising 540 samples, a substantial decrease in error is observed, signifying the model's efficacy even

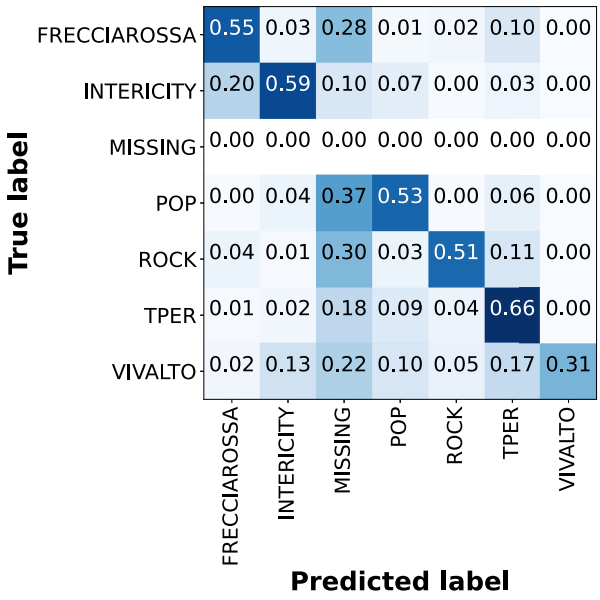


FIGURE 14. Confusion Matrix of the Global Model predictions performances on field dataset.

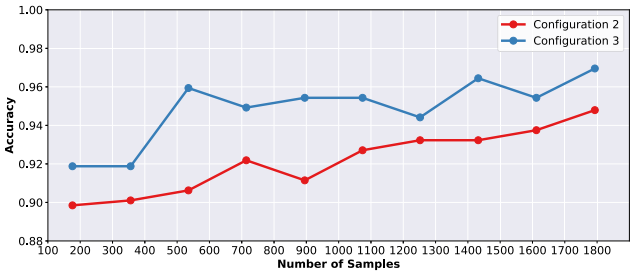


FIGURE 15. Fine-tuning performance on field dataset: comparison of accuracies between configurations 1 and 2.

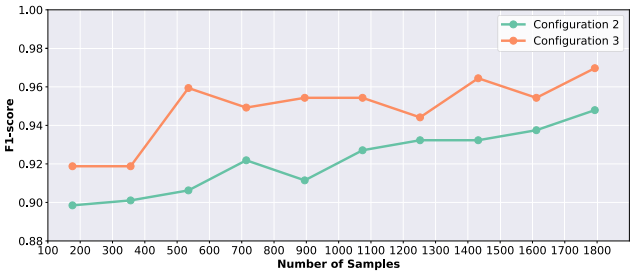
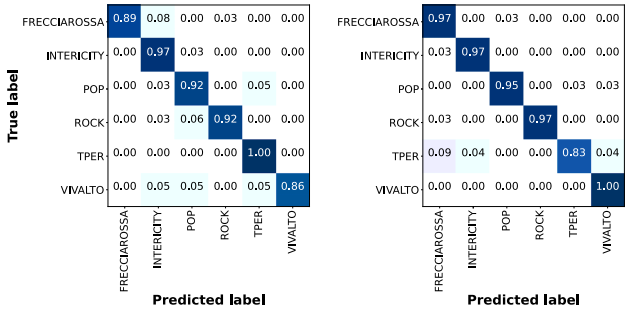


FIGURE 16. Fine-tuning performance on field dataset: F1 scores comparison between configurations 1 and 2.

with limited data. This trend persists as the sample size is increased, reaching a minimum error of 0.69 when the full dataset is employed. As illustrated in Figure 20, a comparison of two levels of fine-tuning shows a pattern with the classification results, indicating that the more fine-tuned model demonstrates higher performance in comparison to its less fine-tuned counterpart.

As demonstrated in Figure 21, the mean error in predicting the number of carriages is shown as a function of the type of train present in the dataset. The model's error is distributed fairly evenly across most labels. However, an exception to this



(a) Configuration 2 (b) Configuration 3

FIGURE 17. Confusion Matrices 540-Samples: on the left (a) in details the classification performance on Linear Fine-Tuning, meanwhile on the right (b) the results on Deep Fine-Tuning.

is observed for the *pop* class, where the error is approximately higher than the other labels. Furthermore, the comparison presented in the figure reinforces this pattern, demonstrating its consistency across both cases.

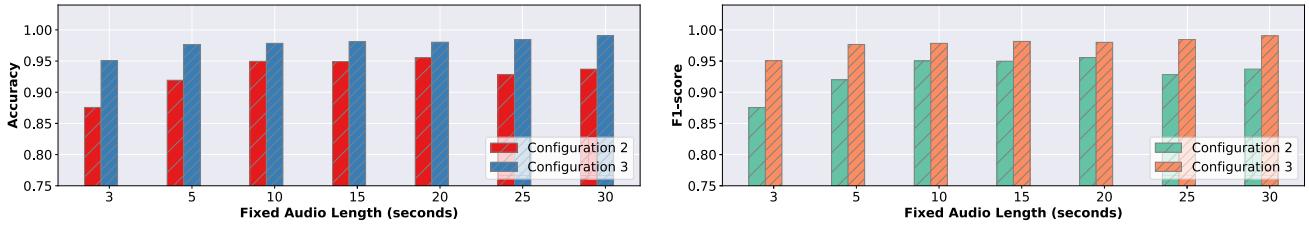
3) SPEED ESTIMATION

The final scenario focused on estimating the speed of the train. Similar to the previous experiments, the same architecture and dataset were used. In addition, the output of the last layer was modified for velocity estimation and the weights of the global model were loaded. As demonstrated in Figure 22, the performance metric demonstrate a decline when compared with the estimation of train carriages, with a MAE of 1.73. However, the error remains minimal, measuring only a few meters per second, even when utilizing 100% of the dataset. It is evident that the model converges after a certain amount of data, indicating that the addition of further data does not enhance overall performance. A comparison of different fine-tuning strategies reveals that deeper fine-tuning consistently results in lower error rates.

E. CLOUD-CONTINUUM PERFORMANCE

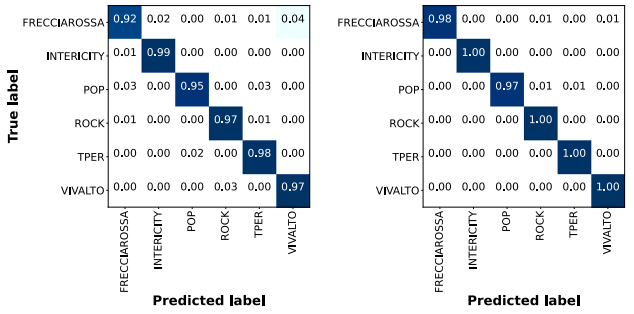
As described in Sec. III, we present an architecture that can be partially or completely deployed on the edge. For this reason, in this section we analyze the performance of the RATTLE framework under various placement configurations between the edge and cloud; We evaluated four deployment configurations:

- **Scenario 1**: Only the data collection is performed on the edge, which transmits the full audio and video payload to the cloud for the subsequent processing steps.
- **Scenario 2**: Data collection, filtering and video feature extraction are executed in the edge, which transmits the data to the cloud for the data analysis. The DL video-based algorithms are compatible with this configuration. It serves as a baseline to be compared with audio classification scenarios.
- **Scenario 3**: Data collection, filtering and mid-term audio features extraction are executed in the edge, which transmits the data to the cloud for analysis. The ML



(a) Accuracy (b) F1-Score

FIGURE 18. Confusion Matrices: (a) Accuracies comparison of fine-tuned models trained on different audio subsets with a fixed duration, and (b) F1-Score comparison of fine-tuned models trained on different audio subsets with a fixed duration.



(a) Configuration 2 (b) Configuration 3

FIGURE 19. Confusion Matrices: (a) Detailed classification performance of Linear Fine-Tuning using data samples with 25% truncation, and (b) results from Deep Fine-Tuning.

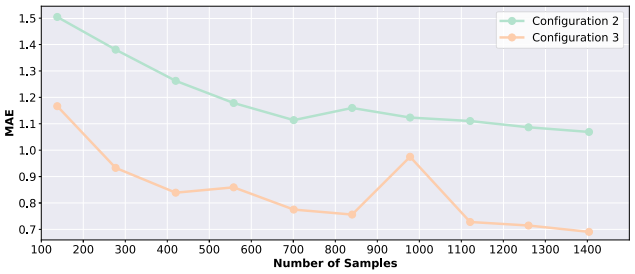


FIGURE 20. Carriage counting: Mean Absolute Error comparison between two different level of fine-tuning on the field dataset.

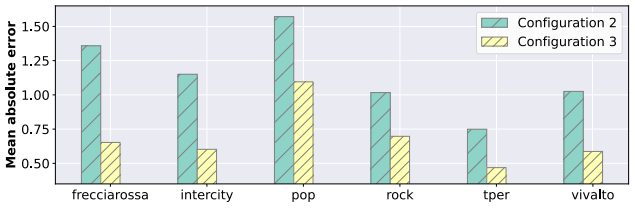


FIGURE 21. Carriage counting MAE per class: prediction error for each class, highlighting performance variations across classes.

algorithms and the FFNN are compatible with this setting.

- **Scenario 4**: Data collection, filtering and mel audio feature extraction are executed in the edge, which transmits the data to the cloud for data analysis. The audio-based DL techniques are compatible with this approach.

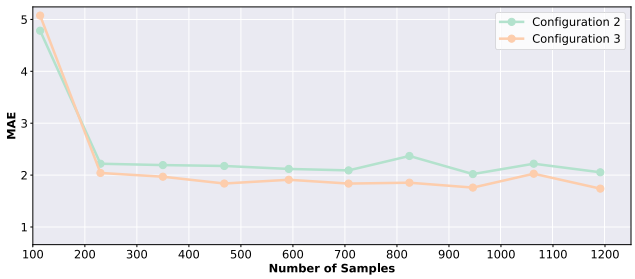


FIGURE 22. Speed estimation: MAE as a function of the number of training samples: impact of sample size on model performance.

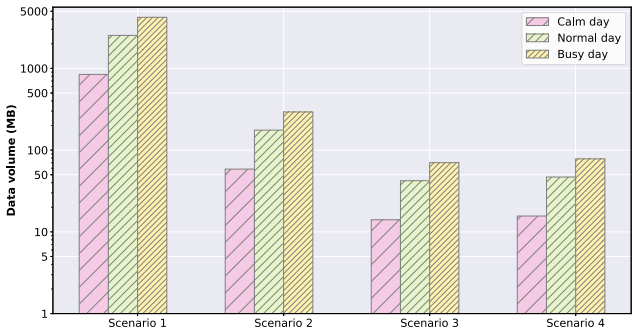


FIGURE 23. Daily cumulative payload size per scenario.

In our test site, the data collection layer captured a minimum of 200 samples per day, an average of 600 samples per day, and a maximum of 1000 samples per day. We defined these values as representing a normal, busy, and calm day, respectively and are based on the actual data distribution of our collected samples. Figure 23 depicts the daily data volume transferred between the edge and the cloud (y-axis is in logarithmic scale) in the four different scenarios considering the three different workloads. Notably, depending on where the pipeline is deployed, the data transfer between edge and cloud nodes varies drastically. In particular, the scenarios in which the audio feature extraction is performed at the edge – Scenario 3 and Scenario 4 – imposed significantly less traffic on the network when compared to the other scenarios. These results motivate the choice of performing train identification through audio, since Scenario 2 that executes video feature extraction at the edge under-performs the other audio-based scenarios. Furthermore, in Section VI-D1 we highlight how

we can constrain the audio sensing length without hindering classification performance, ultimately resulting in *(i)* lower data transfer, *(ii)* decreased computational need for feature extraction, and *(iii)* potentially higher performance thanks to the more standardized input dimension.

F. LIMITATIONS

While the experimental results demonstrate the proposed framework's promise, several limitations must be acknowledged and addressed before reliable production deployment. First, the evaluation does not exhaustively exercise operationally challenging scenarios. Events with multiple trains passing simultaneously or in rapid succession, rare train types under-represented in the training corpus, and freight compositions are yet to address conditions in our framework; these warrant targeted data acquisition and algorithmic adaptation rather than indicating fundamental flaws in the approach. Second, the dataset exhibits class-imbalance and sample-sparsity for specific train categories. Limited examples for infrequent classes constrain the models' ability to learn robust, generalizable representations and increase the risk of misclassification or false negatives. This is a data-regimen issue addressable through prioritized collection, augmentation, and active-learning strategies. Third, extending the framework to estimate train mass in motion requires additional measurement and annotation capabilities. Reliable ground-truth mass labels typically depend on WIM infrastructure and careful calibration; assembling a sufficiently large, accurately annotated WIM-anchored dataset and designing protocols to integrate WIM labels with acoustic and dynamic sensor streams are non-trivial tasks and central to future work.

Although not yet production ready, the framework is suitable for pilot deployments aimed at validation and targeted data collection to remedy the above limitations.

VII. CONCLUSION

In this paper, we present RATTLE, a comprehensive IoT-based framework for TCI using audio fingerprinting. The framework integrates IoT components and a custom DL architecture, CustomNet, covering data acquisition, management, and analytics. Specifically, our contributions include: (1) the design of a continuum-friendly architecture that enables DL-based TCI tasks; (2) the development of a prototype for the RATTLE IoT field system that supports audio recording during train passages, data transmission, and edge-based DL inference; (3) the collection, annotation, and open publication of a dataset comprising video and audio samples for TCI; (4) the creation of custom DL techniques for train classification, carriage counting, and train speed estimation; (5) a comparative analysis of various DL video-based classification techniques (used as baselines) alongside traditional ML and DL audio-based classification methods across multiple datasets; and (6) the exploration of strategies for fine-tuning a general DL model to specific edge locations, including an evaluation of the sample size required for effective adaptation at each location.

Our experimental results demonstrate the effectiveness of the proposed framework: with a dataset comprising 540 training samples, the model achieved a 96% accuracy and F1 score in the classification task, a 0.85 MAE in carriage counting, and a 1.8 MAE in speed estimation through fine-tuning. The results highlight that optimal performance can be achieved with minimal data from newly deployed field nodes and carefully tuned model configurations.

Future work will focus on extending the capabilities of RATTLE in several directions. First, we plan to investigate model quantization techniques to further reduce the model size, making it compatible with extremely constrained microcontrollers and edge devices. Second, we aim to address the aforementioned challenging scenarios, particularly TCI for multiple trains passing simultaneously, which requires advanced signal processing and multi-source audio analysis, as well as expanding coverage to rare train types and freight trains through targeted data collection campaigns. Third, we will explore robustness-oriented fingerprinting strategies inspired by recent advances that address background noise and distortion in open environments. Fourth, we will investigate multimodal integration with vibration-based sensing to study whether audio+vibration fusion can enable weight-related inference in railway contexts. Finally, we plan to conduct pilot deployments in controlled testing environments to validate the framework's performance under diverse

TABLE 6. Train class to variants mapping.

Model	Variants
AGV575	AGV575
ATR1XX	ATR100, ATR110, ATR115, ATR116, ATR120, ATR125
BIPIANO	TAF, TSR
BLUES	HTR312, HTR412
COLLEONE	ATR803
D445	D445
E190	E190
E401	E401
E414	E414
E464	E464.*
E40X	E402, E403
ETR400	ETR400
ETR4XX	ETR485, ETR480, ETR460, ETR463
ETR500	ETR500
ETR675	ETR675
ETR6X0	ETR610, ETR620, ETR630, ETR640, ETR652, ETR675, RABe503
ETR700	ETR700
FLIRT	ETR155, ETR170, ETR330, ETR340, ETR341, ETR342, ETR343, ETR350, ETR360, ETR524, TILO
GIRUNO	RABe501
JAZZ	ETR324, ETR425, ETR526
MINUETTO	MD, ME, ETT, ETR245
ORSETTO	ETI
POP	ETR103, ETR104, ETR204
ROCK	ETR421, ETR521, ETR621
SWING	ATR220
TGV	TGV
VIVALTO	E464.*
XLE	LE
XLN	LN6

operational conditions. These advancements will enhance the scalability, robustness, and utility of the RATTLE framework in real-world railway systems.

APPENDIX

TRAIN GROUPS AGGREGATION BREAKDOWN

See Table 6.

ACKNOWLEDGMENT

The authors extend their gratitude to Akiem483[2] for his valuable contribution to this project. His extensive collection of videos over the years enabled the creation of the global dataset used in their experiments.

REFERENCES

[1] B. Allotta, P. D'Adamio, L. Marini, E. Meli, L. Pugi, and A. Rindi, "A new strategy for dynamic weighing in motion of railway vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3520–3533, Dec. 2015, doi: 10.1109/TITS.2015.2477104.

[2] C. Arcadius Tokognon, B. Gao, G. Y. Tian, and Y. Yan, "Structural health monitoring framework based on Internet of Things: A survey," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 619–635, Jun. 2017, doi: 10.1109/JIOT.2017.2664072.

[3] R. Bajwa, E. Coleri, R. Rajagopal, P. Varaiya, and C. Flores, "Development of a cost-effective wireless vibration weigh-in-motion system to estimate axle weights of trucks," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 32, no. 6, pp. 443–457, Jun. 2017, doi: 10.1111/mice.12269. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12269

[4] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The Internet of Things, fog and cloud continuum: Integration and challenges," *Internet Things*, vols. 3–4, pp. 134–155, Oct. 2018, doi: 10.1016/j.iot.2018.09.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660518300635

[5] N. Bulatovic and S. Djukanovic, "Mel-spectrogram features for acoustic vehicle detection and speed estimation," in *Proc. 26th Int. Conf. Inf. Technol. (IT)*, Feb. 2022, pp. 1–4, doi: 10.1109/IT54280.2022.9743540.

[6] A. Burrello, D. Brunelli, M. Malavisi, and L. Benini, "Enhancing structural health monitoring with vehicle identification and tracking," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2020, pp. 1–6, doi: 10.1109/I2MTC43012.2020.9128641.

[7] E. Cetinic, T. Lipic, and S. Grgic, "Fine-tuning convolutional neural networks for fine art classification," *Expert Syst. Appl.*, vol. 114, pp. 107–118, Dec. 2018, doi: 10.1016/j.eswa.2018.07.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418304421

[8] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token- semantic audio transformer for sound classification and detection," in *Proc. IEEE ICASSP*, May 2022, pp. 646–650, doi: 10.1109/ICASSP43922.2022.9746312.

[9] L. Ciabattini, A. Esposito, L. Sciullo, M. Di Felice, and I. Zyrianoff, "Mel spectrograms of trains' transit for railway monitoring," Zenodo, Jan. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.14780842

[10] L. Ciabattini, L. Sciullo, A. Esposito, I. Zyrianoff, and M. Di Felice, "RATTLE: Train identification through audio fingerprinting," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2024, pp. 47–54, doi: 10.1109/SMARTCOMP61445.2024.00028.

[11] M. Desmond, E. Duesterwald, K. Brimijoin, M. Brachman, and Q. Pan, "Semi-automated data labeling," in *Proc. NeurIPS Competition Demonstration Track*, vol. 133, H. J. Escalante and K. Hofmann, Eds., Jun. 2021, pp. 156–169.

[12] S. Djukanovic, J. Matas, and T. Virtanen, "Acoustic vehicle speed estimation from single sensor measurements," *IEEE Sensors J.*, vol. 21, no. 20, pp. 23317–23324, Oct. 2021, doi: 10.1109/JSEN.2021.3110009.

[13] L. D. Donato, F. Flammini, S. Marrone, C. Mazzariello, R. Nardone, C. Sansone, and V. Vittorini, "A survey on audio-video based defect detection through deep learning in railway maintenance," *IEEE Access*, vol. 10, pp. 65376–65400, 2022, doi: 10.1109/ACCESS.2022.3183102.

[2]https://www.instagram.com/akiem_e483/

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[15] S. Al-Emadi, A. Al-Ali, and A. Al-Ali, "Audio-based drone detection and identification using deep learning techniques with dataset enhancement through generative adversarial networks," *Sensors*, vol. 21, no. 15, p. 4953, Jul. 2021.

[16] L. Gigli, I. Zyrianoff, F. Zonzini, D. Bogomolov, N. Testoni, M. D. Felice, L. De Marchi, G. Augugliaro, C. Mennuti, and A. Marzani, "Next generation edge-cloud continuum architecture for structural health monitoring," *IEEE Trans. Ind. Informat.*, vol. 20, no. 4, pp. 5874–5887, Apr. 2024, doi: 10.1109/TII.2023.3337391.

[17] Á. Incze, H.-B. Jancsó, Z. Szilágyi, A. Farkas, and C. Sulyok, "Bird sound recognition using a convolutional neural network," in *Proc. IEEE 16th Int. Symp. Intell. Syst. Informat. (SISY)*, Sep. 2018, pp. 000295–000300, doi: 10.1109/SISY.2018.8524677.

[18] F. Iqbal, A. Abbasi, A. R. Javed, G. Srivastava, Z. Jalil, and T. R. Gadekallu, "Identification and categorization of unusual Internet of Vehicles events in noisy audio," in *Proc. IEEE 97th Veh. Technol. Conf. (VTC-Spring)*, Jun. 2023, pp. 1–6, doi: 10.1109/VTC2023-SPRING57618.2023.10199432.

[19] C. Kamienski, I. Zyrianoff, L. F. Bittencourt, and M. Di Felice, "IoTinuum: The IoT computing continuum," in *Proc. 20th Int. Conf. Distrib. Comput. Smart Syst. Internet Things (DCOSS-IoT)*, Apr. 2024, pp. 732–739, doi: 10.1109/DCOSS-IOT61029.2024.00112.

[20] N. Kamuni, S. Chintala, N. Kunchakuri, J. S. A. Narasimharaju, and V. Kumar, "Advancing audio fingerprinting accuracy with AI and ML: Addressing background noise and distortion challenges," in *Proc. IEEE 18th Int. Conf. Semantic Comput. (ICSC)*, Feb. 2024, pp. 341–345, doi: 10.1109/ICSC59802.2024.00064.

[21] S. M. Khan, S. Atamturktur, M. Chowdhury, and M. Rahman, "Integration of structural health monitoring and intelligent transportation systems for bridge condition assessment: Current status and future direction," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2107–2122, Aug. 2016, doi: 10.1109/TITS.2016.2520499.

[22] R. Laroca, A. Cidral Boslooper, and D. Menotti, "Automatic counting and identification of train wagons based on computer vision and deep learning," 2020, *arXiv:2010.16307*.

[23] J. Lee, H. Choi, D. Park, Y. Chung, H.-Y. Kim, and S. Yoon, "Fault detection and diagnosis of railway point machines by sound analysis," *Sensors*, vol. 16, no. 4, p. 549, Apr. 2016, doi: 10.3390/s16040549.

[24] A. Maia, A. Boutouchent, Y. Kardjadja, M. Gherari, E. G. Soyak, M. Saqib, K. Boussekar, I. Cilbir, S. Habibi, S. O. Ali, W. Ajib, H. Elbiaze, O. Erçetin, Y. Ghamri-Doudane, and R. Glitho, "A survey on integrated computing, caching, and communication in the cloud-to-edge continuum," *Comput. Commun.*, vol. 219, pp. 128–152, Apr. 2024, doi: 10.1016/j.comcom.2024.03.005.

[25] I. Malik, S. Latif, S. Manzoor, M. Usama, J. Qadir, and R. Jurdak, "Emotions beyond words: Non-speech audio emotion recognition with edge computing," 2023, *arXiv:2305.00725*.

[26] M. Mohaimenuzzaman, C. Bergmeir, and B. Meyer, "Pruning vs XNOR-Net: A comprehensive study of deep learning for audio classification on edge-devices," *IEEE Access*, vol. 10, pp. 6696–6707, 2022, doi: 10.1109/ACCESS.2022.3140807.

[27] A. Mou and M. Milanova, "Performance analysis of deep learning model compression techniques for audio classification on edge devices," *Science*, vol. 6, no. 2, p. 21, 2024.

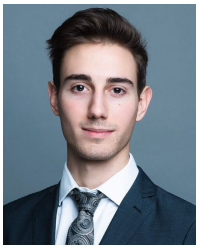
[28] Z. Mushtaq, S.-F. Su, and Q.-V. Tran, "Spectral images based environmental sound classification using CNN with meaningful data augmentation," *Appl. Acoust.*, vol. 172, Jan. 2021, Art. no. 107581, doi: 10.1016/j.apacoust.2020.107581.

[29] Y. Pan, Y. Dong, D. Wang, J. Di, and A. Chen, "Vehicle detection and classification by voiceprint recognition based on single acoustic sensor under bridge expansion joint," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–15, 2024, doi: 10.1109/TIM.2024.3428610.

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[32] A. Shabbir, A. N. Cheema, I. Ullah, I. M. Almanjahie, and F. Alshahrani, "Smart city traffic management: Acoustic-based vehicle detection using stacking-based ensemble deep learning approach," *IEEE Access*, vol. 12, pp. 35947–35956, 2024, doi: 10.1109/ACCESS.2024.3370867.
[33] V.-T. Tran, W.-H. Tsai, Y. Furletov, and M. Gorodnichev, "End-to-end train horn detection for railway transit safety," *Sensors*, vol. 22, no. 12, p. 4453, Jun. 2022, doi: 10.3390/s22124453.
[34] E. Tsalera, A. Papadakis, and M. Samarakou, "Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm," *Energy Rep.*, vol. 6, pp. 223–230, Nov. 2020.
[35] A. Vavilin, A. Lomov, and T. Roman, "Real-time train wagon counting and number recognition algorithm," in *Proc. Int. Workshop Intell. Syst. (IWIS)*, Aug. 2022, pp. 1–4, doi: 10.1109/IWIS56333.2022.9920835.
[36] Y. Wang, Z. Chu, I. Ku, E. C. Smith, and E. T. Matson, "A large-scale UAV audio dataset and audio-based UAV classification using CNN," in *Proc. 6th IEEE Int. Conf. Robotic Comput. (IRC)*, Dec. 2022, pp. 186–189, doi: 10.1109/IRC55401.2022.00039.
[37] K. Zaman, M. Sah, C. Direkoglu, and M. Unoki, "A survey of audio classification using deep learning," *IEEE Access*, vol. 11, pp. 106620–106649, 2023, doi: 10.1109/ACCESS.2023.3318015.
[38] Q. Zhou, J. Shan, W. Ding, C. Wang, S. Yuan, F. Sun, H. Li, and B. Fang, "Cough recognition based on mel-spectrogram and convolutional neural network," *Frontiers Robot. AI*, vol. 8, 2021, Art. no. 580080.
[39] R. Zinno, S. S. Haghshenas, G. Guido, and A. VItale, "Artificial intelligence and structural health monitoring of bridges: A review of the state-of-the-art," *IEEE Access*, vol. 10, pp. 88058–88078, 2022, doi: 10.1109/ACCESS.2022.3199443.



LEONARDO CIABATTINI received the master's degree in management with a major in digital business from HEC Paris, in October 2021. He is currently pursuing the Ph.D. degree, under the supervision of Prof. Marco Di Felice. His master's thesis focused on the impact of technology on the news industry and its distribution channels. He also gained valuable experience in the fields of distributed ledger technology, data management, and software development during his internships at ING Bank and Amazon, and his freelancing activity. His research focuses on the interoperability of the IoT devices and data management on the cloud/edge continuum.

ALFONSO ESPOSITO (Graduate Student Member, IEEE) received the master's degree (summa cum laude) in computer science from the University of Bologna, Italy, in 2023. Title of the thesis "Classificazione e rilevamento del danno in sistemi di monitoraggio strutturale mediante tecniche di machine learning," coordinated by Prof. Marco Di Felice and Prof. Alessandro Marzani. From February 2024 to October 2024, he was a Research Fellow with the I-TROPHYTS PRIN 2022 Project, which involves the design and development of the IoT data analytics solutions for monitoring motor activity in tele-rehabilitation scenarios, with a focus on the development and validation of the IoT and AI solutions to support robotic tele-rehabilitation systems. His current research interests include AI and deep learning, data analytics for structural monitoring and smart health, the IoT, and edge computing.

LUCA SCIULLO received the master's degree (summa cum laude) in computer science from the University of Bologna, Italy, in 2017, where he is currently pursuing the Ph.D. degree in computer science and engineering. He was a Visiting Researcher at the Huawei European Research Center, Munich, Germany. He is a Research Fellow in computer science and engineering at the University of Bologna. He is part of the IoT Prism Laboratory directed by Prof. Marco Di Felice and Prof. Luciano Bononi. His research interests include wireless systems and protocols for emergency scenarios, wireless sensor networks, the IoT systems, and the Web of Things.

IVAN ZYRIANOFF received the B.S. degree in computer science and the M.S. degree in information engineering from the Federal University of ABC, Brazil. His master's thesis encompasses areas, such as the Internet of Things and fog computing. His previous work was in the design, development, and operation of software, and communication components belonging to the IoT Platform for smart irrigation in a Horizon 2020 research project between EU and Brazil.

MARCO DI FELICE received the Laurea (summa cum laude) and Ph.D. degrees in computer science from the University of Bologna, in 2004 and 2008, respectively. He was a Visiting Researcher with Georgia Institute of Technology, Atlanta, GA, USA, and Northeastern University, Boston, MA, USA. He is currently an Associate Professor of computer science with the University of Bologna, where he is the Co-Director of the PRISM IoT Research Laboratory. He authored more than 120 articles on wireless and mobile systems, achieving three best paper awards for his scientific production. He has been the General Chair of IFIP WWIC 2019 conference, the Program Chair of the IEEE Consumer Communications and Networking Conference (IEEE CCNC 2022), the IFIP 16th International Conference on Wired/Wireless Internet Communications (IFIP WWIC 2018), the 2019 5th IEEE International Symposium on Measurements and Networking (IEEE M&N 2019), the IEEE International Workshop on Mission-Oriented Wireless Sensor, UAV and Robot Networking (IEEE MiSARN 2019), the IEEE International Workshop on Wireless Sensor, Robot and UAV Networks (IEEE WiSARN 2018, IEEE WiSARN 2020, and IEEE WiSARN 2021), the International Workshop on Emerging COgnitive Radio Applications and aLgorithms (IEEE CORAL 2012, IEEE CORAL 2013, IEEE CORAL 2015, and IEEE CORAL 2016), and the First International Workshop on Self-Organizing Wireless Access Networks for Smart CITY (IEEE SWANSITY 2014), and the Vice Program Chair of the IEEE Consumer Communications and Networking Conference (IEEE CCNC 2021). He is an Associate Editor of *Ad Hoc Networks* (Elsevier).