

Applying Artificial Intelligence To Classify Urban Noise Sources From Smartphone Recordings: A Small-Scale Pilot Study In Hanoi

Truong Manh Dat*

Faculty of Information Technology, Hanoi University of Natural Resources and Environment
41A Phu Dien Street, Phu Dien Ward, Hanoi, Vietnam

ABSTRACT

Conventional noise monitoring reports sound levels but not their sources, while dedicated sensor networks are costly. This paper presents a low-cost pipeline, from smartphone recording to noise-source classification and web-based monitoring, together with the results of a small-scale pilot in Hanoi. The data comprise 14 GPS-tagged recordings (14.7 minutes in total, 9 recording sessions) covering 8 source classes, including sounds specific to Vietnamese cities such as waste-collection gongs and street-vendor calls. Labels were taken from file names and verified by listening to 111 sample clips, with no labelling error found. Each recording was split in time: the first 70% for training and the last 30% for testing, with data augmentation applied to the training part only. Five models from three families were compared: SVM and Random Forest on MFCC features, a compact CNN on log-Mel spectrograms, and two classifiers on embeddings from the pretrained YAMNet model. On 113 test clips cut from the end of each recording (E1), four models reached accuracies of 0.956–0.991. However, when tested on recordings entirely unseen during training (E2, leave-one-recording-out), the MFCC-based models dropped to 0.363–0.467, whereas the YAMNet-based models retained 0.730–0.779. The CNN trained from scratch did not converge on this amount of data (E1 accuracy 0.204). The YAMNet + Dense model was integrated into a monitoring prototype (FastAPI, SQLite, a browser-based recording page and dashboard). Replaying the 113 test clips through the system API yielded an accuracy of 0.956, identical to the offline result, with a mean processing time of 76.7 ms per clip on a laptop without a GPU. The results show the feasibility of the pipeline and the advantage of transfer learning with little data, and highlight that evaluation within the same recordings is markedly optimistic compared with evaluation on new recordings.

Keywords: urban noise; environmental sound classification; transfer learning; YAMNet; smartphone; data leakage; noise monitoring.

Date of Submission: 09-10-2026

Date of acceptance: 21-10-2026

I. INTRODUCTION

Rapid urbanisation brings denser traffic, more construction and more everyday activity, making noise one of the environmental factors that most visibly affects city residents. The World Health Organization's environmental noise guidelines summarise the evidence linking long-term noise exposure with sleep disturbance, cardiovascular disease and impaired cognition in children [1]. In Vietnam, the National Technical Regulation on Noise QCVN 26:2025/BNNMT sets limit values for five types of area and three periods of the day, and requires measurements to be made with sound level meters that meet the relevant standards [2]. The quantities and procedures for assessing environmental noise are standardised internationally in the ISO 1996 series [3].

The common monitoring practice is campaign-based measurement at fixed points with a sound level meter. The results are indicators such as the equivalent continuous level L_{Aeq} , which show whether a limit is exceeded but not where the noise comes from, although this information is needed to choose mitigation measures. The SONYC project in New York is a well-known example of combining low-cost acoustic sensor networks with machine learning to identify noise sources [4], but it still requires installation, power, communication and maintenance infrastructure. Another direction is to use smartphones as mobile sensors: NoiseTube allowed citizens to measure and map noise with their phones [5]; later evaluations showed that smartphone sound-measurement apps reach acceptable errors only after calibration, and that accuracy depends on the device [6], [7].

Environmental sound classification has advanced rapidly thanks to public datasets such as UrbanSound8K [8] and ESC-50 [9], and to convolutional neural networks (CNNs) operating on spectral representations [10], [11]. The large-scale AudioSet corpus [12] made it possible to pretrain models such as YAMNet [13] and PANNs [14], whose intermediate representations (embeddings) can be reused for new tasks with little data. In Vietnam, urban noise studies have mainly focused on measuring current levels and comparing them with regulations [15]; applications of artificial intelligence to identify noise sources from locally collected data remain limited. Some characteristic sources, such as gongs announcing waste collection or the calls of street vendors, are absent from international datasets.

An important methodological issue with small, field-collected datasets is how the training and test sets are split. Clips cut from the same recording share almost identical characteristics (same place, time and background); if they appear in both sets, the evaluation overestimates performance. The problem becomes worse when data augmentation is applied before splitting [16].

This paper reports the results of a university-level research project carried out with limited resources (one researcher, one smartphone, one laptop). The contributions are: (i) a smartphone data-collection procedure in which the label, time and coordinates are encoded directly in the file name, followed by a label check by listening; (ii) a pilot dataset of 8 noise-source classes in Hanoi; (iii) a comparison of five models under three evaluation schemes (time split within recordings, leave-one-recording-out, and a random-split control), quantifying how optimistic lenient evaluations are; (iv) a prototype monitoring system running on commodity hardware, verified by replaying test data. Section 2 describes the data and methods, Section 3 presents and discusses the results, and Section 4 concludes.

II. DATA AND METHODS

The overall workflow is shown in Figure 1. It consists of three stages corresponding to the tasks of the project: data collection and preparation; building and evaluating classification models; and integrating the model into a monitoring system.

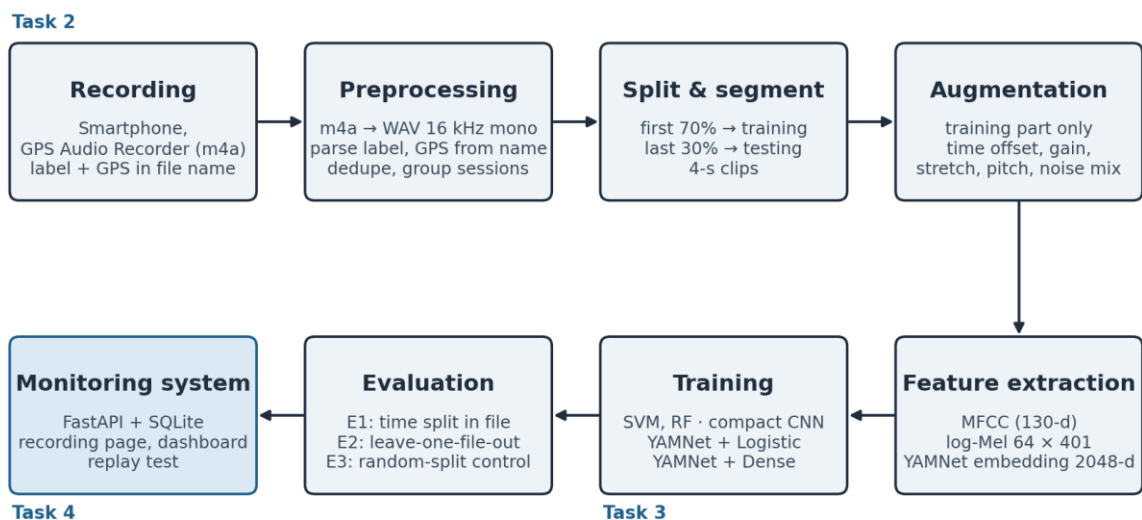


Figure 1. Overall research workflow

2.1. Recording sites and equipment

Data were recorded at locations with typical noise sources in several parts of Hanoi: the Phu Dien area (near Hanoi University of Natural Resources and Environment), a section of the Nhon – Hanoi Station elevated urban railway, the Nghia Do area, and the national railway line passing through the city centre north of Hanoi Station. The recording points span about 9 km from east to west (Figure 2).

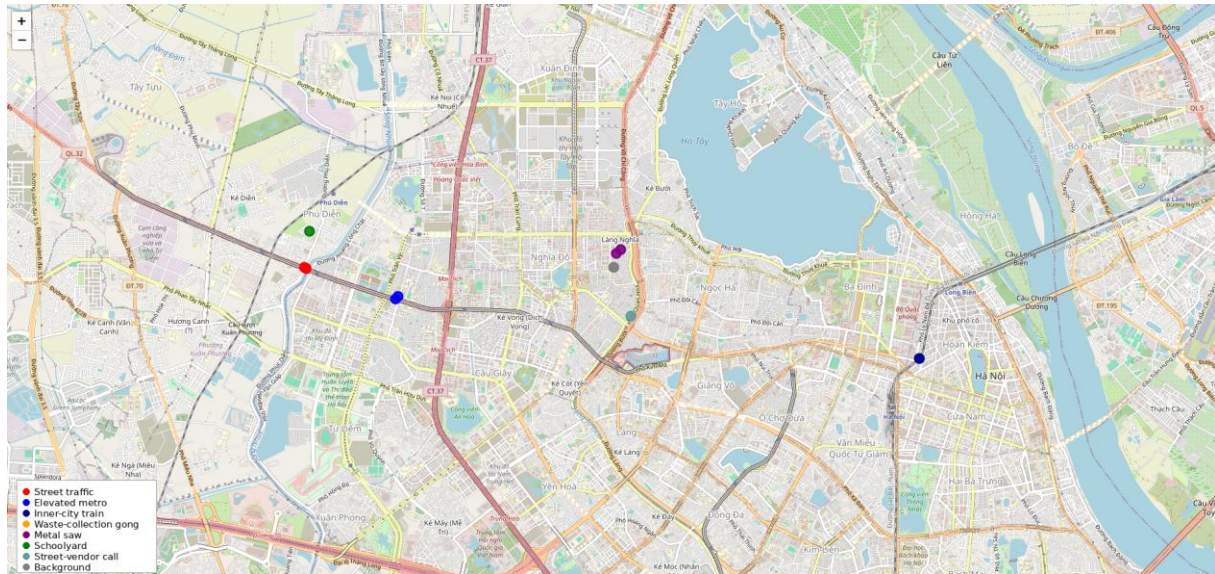


Figure 2. Recording locations (GPS coordinates stored in the file names; base map © OpenStreetMap contributors)

The recording device was a Samsung Galaxy S24 smartphone running the GPS Audio Recorder app, which saves audio in m4a format (AAC encoding). All data were recorded with the same device. Immediately after each recording, the file was named following the pattern “<source name> - Recording_<date>_<time>_GPS_<latitude>_<longitude>.m4a”; the label, time and coordinates (four decimal places) are therefore stored in the file name and parsed automatically during preprocessing.

2.2. Noise-source classes

Eight classes were selected based on the sources commonly encountered in the surveyed areas (Table 1). Besides classes found in international datasets (traffic, trains, mechanical tools, schoolyards), the dataset includes two classes specific to Vietnamese cities: waste-collection gongs and street-vendor calls.

Table 1. Noise-source classes in the dataset

Class	Label code	Description
Street traffic	duong_pho	Engine, tyre and horn noise from motorbikes and cars on roads
Elevated metro	tau_dien	Urban railway trains running on the elevated viaduct
Inner-city train	tau_hoa	Trains on the national railway line passing through inner-city residential areas
Waste-collection gong	keng_rac	Metal gong struck to announce household waste collection; repetitive impulsive sound
Metal saw	may_cua	Metal saws and metal-cutting machines
Schoolyard	san_truong	Pupils talking, laughing and playing in a schoolyard
Street-vendor call	rao_hang	Calls of street vendors (human voice, possibly played through a loudspeaker)
Background	nen	Periods without a dominant noise source

2.3. Preprocessing and label verification

The m4a files were converted to mono WAV at 16 kHz with FFmpeg. Labels were derived from the file names using a keyword table (after removing Vietnamese diacritics), e.g. “Âm thanh đường phố” (street sound) → duong_pho. Duplicate files were detected by MD5 hash and by identical recording time; in this way a file named “Tàu điện vào ga” (metro entering the station) that had the same recording time as an “elevated metro” file was detected and removed, leaving 14 files. Files recorded on the same day less than 200 m apart were grouped into one recording session.

Because a label is assigned to a whole file by its name, a verification step was added: for each class, up to 15 original 4-s clips were randomly selected (the metal-saw class has only 6 original clips), and the researcher listened to them and flagged any clip whose label was wrong. In total, 111 clips were checked, and none was flagged as mislabelled.

2.4. Data splitting, segmentation and augmentation

Since most classes have only one or two files, data were split in time within each file (Figure 3). After discarding 1 s at the beginning and end of each file (button-press sounds), the last 30% of the duration was reserved for testing and the first part for training, with a 2-s gap between them so that no clip contains audio from both parts. The training part was cut into 4-s clips with a 1-s hop; the test part was cut with a 2-s hop.

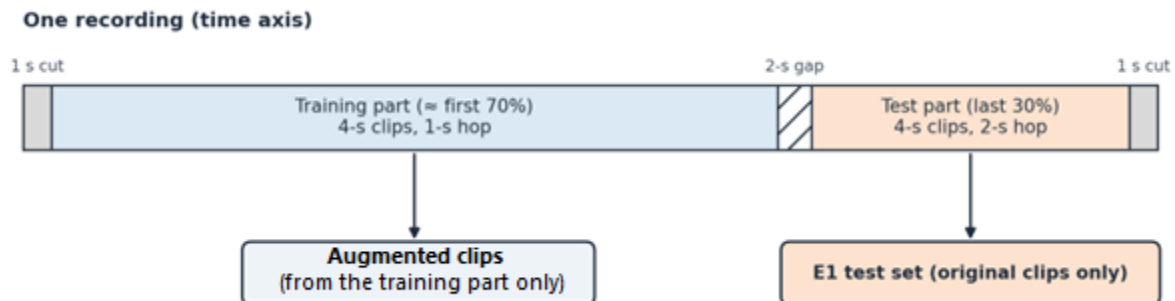


Figure 3. Time-based split of one recording into a training part and a test part

Because the number of original clips differs greatly between classes, classes with fewer than 200 training clips were augmented up to 200. Each augmented clip was cut at a random start point within the training part of a file, and the following transformations were applied at random (probability 0.6 each): time stretching by a factor of 0.92-1.08, pitch shifting by ± 1 semitone, mixing with a background clip (taken from the training part of the background class) at a signal-to-noise ratio of 8-20 dB, and gain change of ± 6 dB [11]. Crucially, augmentation was performed **only** on the training part and **after** the test part had been separated, so no augmented clip contains audio from the test part.

2.5. Feature extraction

MFCC features. Mel-frequency cepstral coefficients (MFCCs) [17] are computed from the energies of triangular filters spaced on the Mel scale, defined by Eq. (1); the n -th coefficient is given by Eq. (2):

$$m = 2595 \cdot \log_{10}(1 + f/700) \quad (1)$$

$$c_n = \sum_{m=1 \dots M} \log(S_m) \cdot \cos[\pi n(m - 0.5)/M], \quad n = 1, \dots, N \quad (2)$$

where f is frequency (Hz), S_m is the output energy of the m -th Mel filter and M is the number of filters. For each 4-s clip, 40 MFCCs were computed per frame; their mean and standard deviation over time, the standard deviation of the delta coefficients, and the mean and standard deviation of five spectral descriptors (spectral centroid, bandwidth, roll-off, zero-crossing rate and RMS energy) form a 130-dimensional vector.

Log-Mel spectrogram. A short-time Fourier transform with a 25-ms window and 10-ms hop was projected onto 64 Mel bands and log-compressed; each clip is represented as a 64×401 matrix.

YAMNet embeddings. YAMNet is an audio classifier based on the MobileNetV1 architecture [18], pretrained on AudioSet with 521 event classes [12], [13]. It splits a 16-kHz signal into 0.96-s frames with a 0.48-s hop and produces a 1024-dimensional embedding per frame. For each 4-s clip, the mean and maximum of the frame embeddings over time were concatenated into a 2048-dimensional vector. MFCC and log-Mel features were computed with librosa [19].

2.6. Classification models

Group M1 – classical machine learning on the MFCC vector: (M1a) SVM with an RBF kernel, $C = 10$, on standardised data; (M1b) Random Forest with 300 trees. Both were implemented with scikit-learn [20].

Group M2 – compact CNN on log-Mel spectrograms: three convolutional blocks (32, 64 and 128 filters of size 3×3 , batch normalisation, ReLU, 2×2 max pooling), global average pooling, dropout 0.3 and a softmax layer. The network was trained from scratch with Adam (learning rate 10^{-3} , batch size 32, up to 40 epochs, early stopping after 6 epochs without improvement on 15% of the training data held out for validation).

Group M3 – transfer learning with YAMNet. YAMNet weights were frozen and used only as a feature extractor. On the standardised 2048-dimensional embeddings, two classifiers were trained: (M3a) logistic regression; (M3b) a fully connected network with one hidden layer of 256 units (ReLU), dropout 0.4 and a softmax output, trained for 60 epochs with Adam.

All models used class weights inversely proportional to class frequency. Experiments were run on a Lenovo ThinkBook 15 G2 ITL laptop (11th-generation Intel Core CPU, 16 GB RAM, no GPU) with Python 3.10, TensorFlow 2.17 and scikit-learn.

2.7. Evaluation

Three evaluation schemes were used:

- **E1 - time split within recordings (main evaluation):** train on the training parts (original and augmented clips) of all files and test on the 113 original clips of the test parts. E1 covers all 8 classes, but test and training clips come from the same files, so the result reflects the ability to recognise a later portion of an already known acoustic situation.
- **E2 - leave-one-recording-out:** for each file of a class with at least two files, train on all clips of the other files (including augmented clips generated from those files) and test on all original clips of the held-out file. E2 involves 12 files from 6 classes (the schoolyard and street-vendor classes are excluded because each has a single file), with 366 test clips in total. This scheme is closest to encountering a new recording. The CNN was not evaluated with E2 because it would have to be retrained 12 times.
- **E3 - random-split control:** pool all clips (original and augmented) and split them randomly 70/30. This is deliberately methodologically wrong and is used only to illustrate how much results are inflated when splitting after augmentation.

The metrics are accuracy, per-class precision P_k , recall R_k and FI_k , and macro-F1, the mean of FI_k over K_c classes:

$$P_k = TP_k / (TP_k + FP_k), \quad R_k = TP_k / (TP_k + FN_k) \quad (3)$$

$$FI_k = 2P_k R_k / (P_k + R_k), \quad \text{macro-F1} = (1/K_c) \cdot \sum_k FI_k \quad (4)$$

where TP_k , FP_k and FN_k are the numbers of true positives, false positives and false negatives of class k . For E2, the mean recall over the 6 participating classes is reported in addition to accuracy.

2.8. Noise monitoring system

The system architecture is shown in Figure 4. The recording page is a web page running in a browser (phone or computer) that uses the Web Audio API to record continuous 4-s clips, converts them to 16-kHz WAV and sends them to the server together with GPS coordinates; users can press buttons to record the source they observe, as a reference. The analysis service, written with FastAPI, receives each clip, extracts the YAMNet embedding, predicts the label with model M3b and estimates a relative A-weighted level. Results are stored in an SQLite database and displayed on a dashboard (Leaflet, Chart.js) with a map, level over time, source shares and a per-session confusion matrix. The whole system runs on the same laptop used for training.

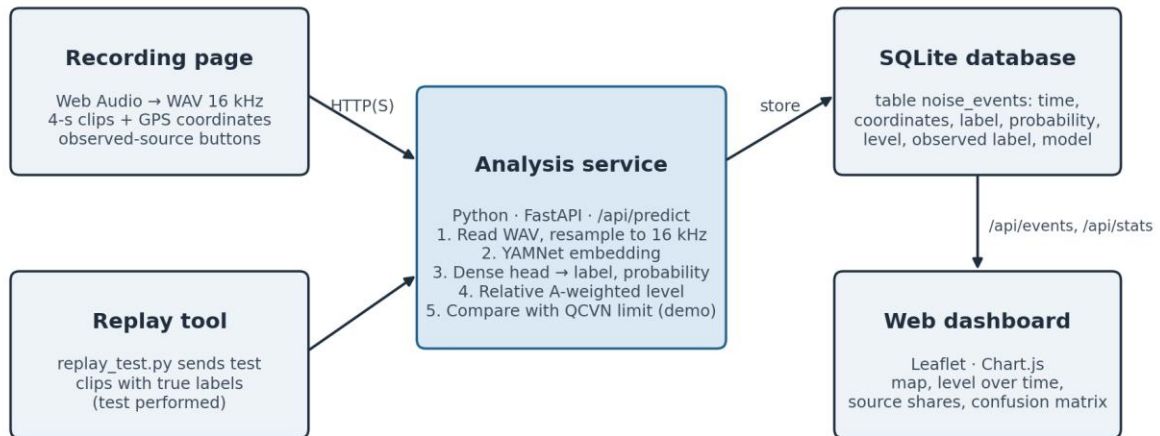


Figure 4. Architecture of the prototype noise monitoring system

The A-weighted level of each clip is computed in the frequency domain by Eq. (5), and the equivalent level of a session by Eq. (6):

$$L_A = 10 \cdot \log_{10} [\sum_k W_A(f_k) \cdot |X(f_k)|^2 / E_0] + C \quad (5)$$

$$L_{Aeq} = 10 \cdot \log_{10} [(1/J) \cdot \sum_{j=1 \dots J} p_j], \quad p_j = 10^{0.1 \cdot L_{A,j}} \quad (6)$$

where $X(f_k)$ is the spectrum of the clip normalised to digital full scale, $W_A(f)$ is the A-weighting curve (dB), $L_{A,j}$ is the level of clip j , E_0 is a normalisation constant, J is the number of clips in the session and C is the calibration constant. As no reference sound level meter was available, $C = 0$; the level is therefore relative to digital full scale (usually negative), suitable only for comparing clips with each other and not for comparison with the limits of QCVN 26:2025/BNNMT.

System test. To verify the whole processing chain (file reading, feature extraction, prediction, database storage, aggregation), a version of model M3b trained only on the training parts (as in E1) was loaded into the server. A replay tool sent the 113 E1 test clips one by one to the /api/predict endpoint together with their true labels; accuracy, the confusion matrix and processing time were then aggregated directly from the database and compared with the offline result of the same model. Owing to the constraints of the project, no live on-site recording test was carried out and the sound level was not calibrated.

III. RESULTS AND DISCUSSION

3.1. Dataset

The dataset consists of 14 files recorded in 9 sessions between 23 April 2026 and 3 October 2026; file durations range from 12.9 s to 300.8 s, for a total of 880 s (14.7 minutes). Per-class statistics are given in Table 2.

Table 2. Dataset statistics by class

Class	Files	Sessions	Duration (s)	Original training clips	Augmented clips	Test clips (E1)
Street traffic	2	1	242	155	45	34
Elevated metro	2	1	50	22	178	5
Inner-city train	2	1	74	39	161	8
Waste-collection gong	2	2	45	18	182	4
Metal saw	2	1	27	4	196	2
Schoolyard	1	1	301	203	0	44
Street-vendor call	1	1	33	16	184	3
Background	2	1	108	62	138	13
Total	14	9	880	519	1,084	113

The dataset is very small and imbalanced. Except for the waste-collection gong (2 sessions), each class was recorded in a single session; the schoolyard and street-vendor classes have only one file each. The metal-saw class contains only 27 s of audio, giving 4 original training clips and 2 test clips; its 196 augmented clips were all generated from about 11 s of training audio, so their real diversity is very low. In total there are 1,716 clips (519 original training clips, 1,084 augmented clips and 113 test clips). Extracting YAMNet embeddings for all of them took about 0.5 minutes on the CPU.

3.2. Model comparison

Table 3 and Figure 5 present the results of the five models under the three evaluation schemes.

Table 3. Classification results of the models under three evaluation schemes

Model	E1 accuracy	E1 macro-F1	E2 accuracy	E2 mean recall (6 classes)	E3 macro-F1 (control)	E1 training time (s)
M1a - SVM (MFCC)	0.956	0.977	0.467	0.579	1.000	0.06
M1b - Random Forest (MFCC)	0.991	0.997	0.363	0.542	0.998	1.1
M2 - Compact CNN (log-Mel)	0.204	0.219	-	-	0.120	191.2
M3a - YAMNet + Logistic	0.956	0.984	0.779	0.839	0.998	0.34
M3b - YAMNet + Dense	0.956	0.956	0.730	0.817	0.998	16.5

Note: E1 - 113 test clips; E2 - 366 clips from 12 files of 6 classes; “-”: not evaluated. Training time on CPU, excluding feature extraction.

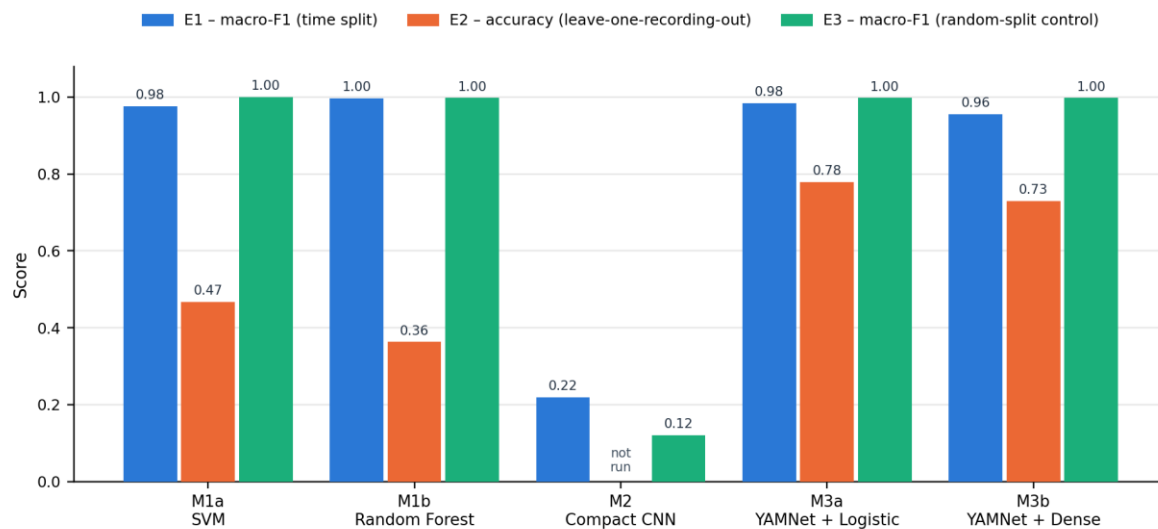


Figure 5. Comparison of the models under the three evaluation schemes

Under E1, four models (M1a, M1b, M3a, M3b) reached accuracies between 0.956 and 0.991; Random Forest was highest (0.991, 1 error out of 113 clips). However, with 113 test clips each clip corresponds to almost 0.9 percentage points, so the differences between these models amount to only 0-4 clips and are not sufficient to rank them.

E2 gives a very different picture. When tested on recordings not used for training, SVM accuracy fell from 0.956 to 0.467 and Random Forest from 0.991 to 0.363, meaning that these two models mostly memorised file-specific characteristics (background, location, time) rather than properties of the noise source. The YAMNet-based models degraded much less: M3a retained 0.779 (mean recall 0.839) and M3b 0.730 (0.817). Representations learned from the large-scale AudioSet corpus thus generalise to new recordings clearly better than MFCC features when training data are very scarce, in line with [13], [14]. Note that E2 covers only 6 classes and that, for many classes, the two files were recorded in the same session, so even E2 may still be optimistic compared with other places and times.

The compact CNN trained from scratch did not converge: its E1 accuracy was only 0.204 and its E3 macro-F1 only 0.120, low even under the lenient random split. With about 1,600 training clips, most of them augmented from a few tens of seconds of audio, the data are insufficient to train a convolutional network from scratch. This result does not reflect the capability of CNNs in general; rather, it shows that with very little data, reusing a pretrained model is the more suitable choice. Notably, the CNN's training cost (191 s) was much higher than that of M3a (0.34 s) and M3b (16.5 s).

The E3 results (macro-F1 0.998-1.000 for the M1 and M3 models) are almost perfect, showing that when data are augmented first and then split at random, near-duplicate clips appear in both sets and the results are inflated. For the same model, E3 macro-F1 exceeds E2 accuracy by between 0.22 (M3a) and 0.64 (M1b). For this reason, E3 is not used to draw conclusions, and we recommend that studies with small field-recorded datasets split the data by collection unit (recording, session or location) [16].

The model used in the system is M3b, which was chosen at the design stage. According to E2, M3a is 0.049 more accurate than M3b and trains faster; with the current amount of data this difference is not conclusive, but M3a is a candidate worth considering when the dataset is extended.

3.3. Per-class results

Table 4 gives the per-class E1 results of M3b (the model used in the system), together with the F1 of Random Forest for comparison.

Table 4. Per-class E1 results

Class	Test clips	M3b precision	M3b recall	M3b F1	M1b F1	Reliability
Street traffic	34	0.919	1.000	0.958	0.986	1 session
Elevated metro	5	1.000	1.000	1.000	1.000	1 session
Inner-city train	8	1.000	1.000	1.000	1.000	1 session
Waste-collection gong	4	1.000	1.000	1.000	1.000	< 5 clips

Class	Test clips	M3b precision	M3b recall	M3b F1	M1b F1	Reliability
Metal saw	2	1.000	1.000	1.000	1.000	< 5 clips, 1 session
Schoolyard	44	1.000	0.886	0.940	0.989	1 file
Street-vendor call	3	0.600	1.000	0.750	1.000	< 5 clips, 1 file
Background	13	1.000	1.000	1.000	1.000	1 session

By the criterion adopted in this study, results for a class with fewer than 5 test clips or a single recording session are considered unreliable; all 8 classes fall into this category, and the values of 1.000 in Table 4 only mean that the model recognised all of the few available test clips, not that it is perfectly accurate. All 5 errors of M3b concern the schoolyard class: 3 clips were labelled street traffic and 2 street-vendor call (Figure 6). Acoustically, schoolyard sound consists of many overlapping voices, possibly mixed with traffic from nearby roads, and is therefore easily confused with traffic and with vendor calls (also human voice). Random Forest misclassified only 1 schoolyard clip (as street traffic) in E1, but, as discussed in Section 3.2, this advantage does not hold for new recordings.

3.4. System test results

The results of replaying the 113 test clips through the system are given in Table 5 and Figure 6.

Table 5. Replay test through the monitoring system (model M3b, E1 version)

Indicator	Result
Number of 4-s clips sent to the /api/predict endpoint	113
System accuracy	0.956 (108/113 clips)
System macro-F1	0.956
Accuracy of the same model run offline	0.956 (113 clips)
Server processing time: mean / median / 95th percentile	76.7 / 73.5 / 85.8 ms
Sound level calibration	Not performed (relative level)

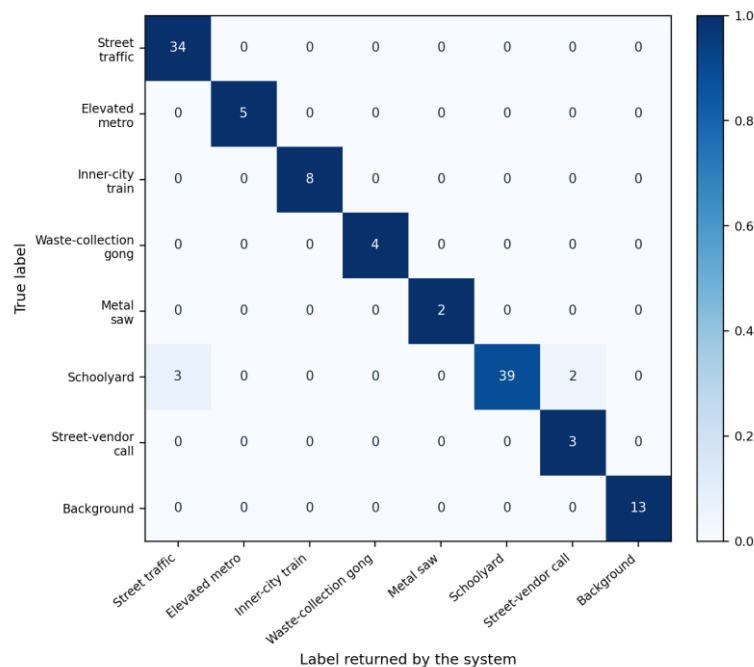


Figure 6. Confusion matrix when replaying the 113 test clips through the system (rows: true label; columns: label returned by the system)

The system accuracy matches the offline result of the same model exactly (0.956 on 113 clips), showing that the processing chain inside the system (file reading, normalisation, embedding extraction, prediction, storage, aggregation) is consistent with the experimental environment. The mean processing time of 76.7 ms for a 4-s clip,

less than 2% of the clip length, on a laptop without a GPU indicates that, computationally, a commodity server could handle many recording sources at once. Figure 7 shows the recording page and the dashboard.

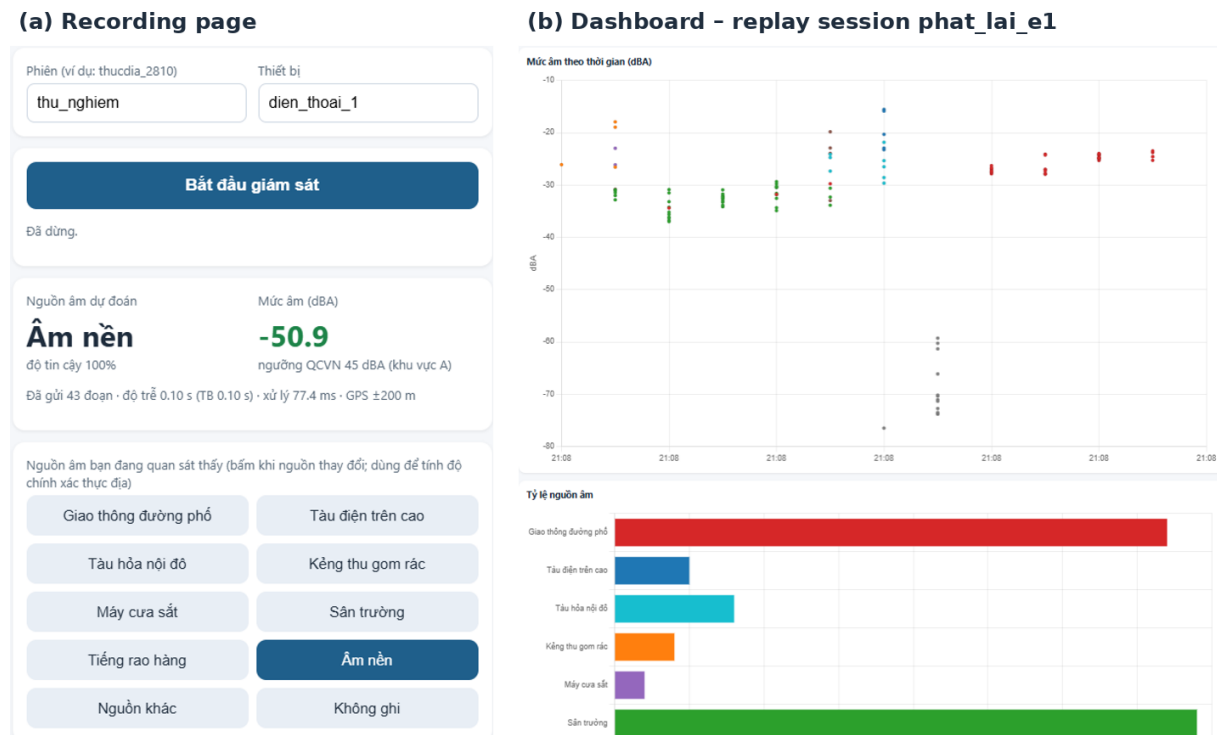


Figure 7. System interface (in Vietnamese): (a) recording page (screenshot taken during functional testing; values shown are for illustration only); (b) dashboard of the replay session

It should be stressed that the replay test verifies the correctness of the software, not recognition performance under real operating conditions: the data were sent as digital files, did not pass through a loudspeaker and microphone, contained no sources outside the 8 classes, and the end-to-end network latency could not be measured. The level shown on the dashboard is relative (for example, the equivalent level of the replay session is -26.1 dB relative to full scale), so the QCVN exceedance indicator on the interface is not yet meaningful.

3.5. Discussion

The results show that the proposed pipeline can be implemented end to end at almost zero hardware cost: data were recorded with a personal phone, naming files according to a template replaced a dedicated annotation tool, models were trained on a laptop without a GPU (extracting YAMNet embeddings for the whole dataset took about 0.5 minutes) and the monitoring system runs on open-source tools. The most important methodological finding is the large gap between E1 and E2: with small field-recorded datasets, evaluation within the same recordings gives almost perfect results that do not reflect the ability to recognise new recordings. Under these conditions, transfer learning from a pretrained model was the most suitable of the approaches tested.

The study has limitations that should be stated clearly. First, the dataset is very small (14 files, 14.7 minutes); each class has only 1-2 files and most come from a single session, so the figures in this paper are preliminary and cannot be generalised. Second, E1 uses test clips from the same files as the training clips; evaluation on independent files (E2) was possible for only 6 classes. Third, the data were recorded with a single device, labels were assigned per file by name and checked by only one person. Fourth, the system has only been verified by replaying data; no live on-site recording test has been performed. Finally, the phone microphone has not been calibrated against a reference sound level meter, so levels are only relative. Under QCVN 26:2025/BNNMT, compliance assessment must use measuring instruments that meet the relevant standards [2]; the proposed system is therefore only a support tool for identifying noise sources and does not replace official monitoring.

IV. CONCLUSION

This paper has presented a low-cost pipeline that applies artificial intelligence to classify urban noise sources, from smartphone recordings with GPS coordinates to a web-based monitoring system, together with experimental results on an 8-class dataset of 14 recordings from Hanoi. With a time split within recordings, four

models reached accuracies of 0.956-0.991; when tested on new recordings, the models based on YAMNet embeddings (0.730-0.779) clearly outperformed those based on MFCCs (0.363-0.467), and the CNN trained from scratch did not converge. The monitoring system integrating the YAMNet + Dense model reproduced the offline result exactly in the replay test (0.956 on 113 clips), with a processing time of 76.7 ms per clip on commodity hardware.

Future work includes: extending the data with more sessions, locations and devices per class to allow evaluation by session and by location; adding an “other source” class and moving to multi-label classification; live on-site tests with an observer recording the actual sources to measure real accuracy and latency; calibrating the phone microphone against a reference sound level meter; and re-assessing the choice between M3a and M3b on the extended dataset.

Acknowledgements

This research was carried out within the university-level science and technology project “Research on the application of artificial intelligence in urban noise assessment and monitoring”, code [HUNRE.2026.11.02], Hanoi University of Natural Resources and Environment.

REFERENCES

- [1] World Health Organization (2018). *Environmental Noise Guidelines for the European Region*. WHO Regional Office for Europe, Copenhagen.
- [2] Ministry of Agriculture and Environment of Vietnam (2025). *QCVN 26:2025/BNNMT – National Technical Regulation on Noise*, issued with Circular No. 01/2025/TT-BNNMT dated 15 May 2025 (in Vietnamese).
- [3] ISO (2016). *ISO 1996-1:2016 Acoustics – Description, measurement and assessment of environmental noise – Part 1: Basic quantities and assessment procedures*. International Organization for Standardization, Geneva.
- [4] Bello, J. P., Silva, C., Nov, O., Dubois, R. L., Arora, A., Salamon, J., Mydlarz, C., Doraiswamy, H. (2019). SONYC: A system for monitoring, analyzing, and mitigating urban noise pollution. *Communications of the ACM*, 62(2), 68–77.
- [5] Maisonneuve, N., Stevens, M., Niessen, M. E., Steels, L. (2009). NoiseTube: Measuring and mapping noise pollution with mobile phones. In *Information Technologies in Environmental Engineering*, Springer, Berlin, Heidelberg, 215–228.
- [6] Kardous, C. A., Shaw, P. B. (2014). Evaluation of smartphone sound measurement applications. *The Journal of the Acoustical Society of America*, 135(4), EL186–EL192.
- [7] Murphy, E., King, E. A. (2016). Testing the accuracy of smartphones and sound level meter applications for measuring environmental noise. *Applied Acoustics*, 106, 16–22.
- [8] Salamon, J., Jacoby, C., Bello, J. P. (2014). A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM International Conference on Multimedia*, 1041–1044.
- [9] Piczak, K. J. (2015). ESC: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, 1015–1018.
- [10] Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In *Proceedings of the IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6.
- [11] Salamon, J., Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3), 279–283.
- [12] Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., Ritter, M. (2017). Audio Set: An ontology and human-labeled dataset for audio events. In *Proceedings of IEEE ICASSP 2017*, 776–780.
- [13] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., et al. (2017). CNN architectures for large-scale audio classification. In *Proceedings of IEEE ICASSP 2017*, 131–135.
- [14] Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., Plumbley, M. D. (2020). PANNs: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, 2880–2894.
- [15] Nguyen Ngoc Anh, Bui Thi Thanh Huong, Hoang Gia Trang (2022). Noise pollution in schools of Hanoi city: Status and solutions. *VNU Journal of Science: Education Research*, 39(1). DOI: 10.25073/2588-1159/vnuer.4648.
- [16] Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillerá-Arroita, G., et al. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, 40(8), 913–929.
- [17] Davis, S., Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 357–366.
- [18] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- [19] McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., Nieto, O. (2015). librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*, 18–25.
- [20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.