Sensor Applications

TinyML Models for a Low-cost Air Quality Monitoring Device

I Nyoman Kusuma Wardana[1,3*], Julian W. Gardner[1**], and Suhaib A. Fahmy[2***]

[1] *School of Engineering, University of Warwick, Coventry CV4 7AL, UK*
[2] *King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia*
[3] *Department of Electrical Engineering, Politeknik Negeri Bali, Badung, 80364, Bali, Indonesia*
[*] *Graduate Student Member, IEEE*
[**] *Fellow, IEEE*
[***] *Senior Member, IEEE*

Manuscript received Month Date, 2023; revised Month Date, 2023; accepted Month Date, 2023. Date of publication Month Date, 2023; date of current version Month Date, 2023.

Abstract—Low-cost air quality monitoring devices can provide high-density spatiotemporal pollution data and thus offer a better opportunity to apply machine learning. Low-cost sensor nodes usually utilize microcontrollers as the main processors, and tinyML brings machine learning (ML) models to these resource-constrained devices. In this letter, we reported the development of a low-cost air quality monitoring device with embedded tinyML models. We deployed two tinyML models on a single microcontroller and performed two tasks: predicting air quality and power parameters (using model predictor) and imputing missing features (using model imputer). The proposed model predictor can estimate parameters with a coefficient of determination above 0.70, and the model imputer effectively estimates the testing data when missing rates are below 80%. By performing the post-training quantization technique, we can further reduce the model size but slightly degrade the accuracies.

Index Terms—TinyML, air quality prediction, missing data, microcontrollers, low-cost devices.

I. INTRODUCTION

Recent research has demonstrated the feasibility of low-cost sensor nodes for air quality monitoring systems [1]–[3]. This emerging sensor-based air quality monitoring field can provide high-density spatiotemporal pollution data, supplementing the established methodology with more precise and expensive devices [4]. The immense volume of collected spatiotemporal data has provided a better opportunity to apply machine learning (ML) techniques in air quality areas, such as air contaminant prediction [5], [6], missing data imputation [7], [8], and classification tasks [9].

TinyML is a cutting-edge field of artificial intelligence. It brings machine learning (ML) models to resource-constrained devices, such as microcontrollers [10]. A microcontroller is typically limited to its memory and computational capabilities. Effective deployment of tinyML models requires a thorough understanding of hardware, software, algorithms, and applications. Regardless of their limited performance, microcontrollers can gather physical environment data through sensors and perform decisions based on ML algorithms [11].

The key motivation of this letter is to empower a low-cost air quality device with intelligence. We deployed two different tinyML models to a single microcontroller. One model is used to predict air status and electrical power parameters, whereas another is employed to impute missing air pollution data. To the best of our knowledge, previous works on air quality prediction using tinyML have not specifically explored prediction and imputation tasks on a single microcontroller.

Corresponding author: INK Wardana (e-mail: kusuma.wardana@warwick.ac.uk)
Associate Editor: First Last.
Digital Object Identifier xx.xxxx/LSENS.2023.0000000

II. METHOD

A. Dataset and Preprocessing

A dataset obtained from a direct measurement is used to train and evaluate the tinyML models. The air quality monitoring device was used to collect data over approximately three months, from 21 July 2022 to 20 October 2022. Eight features were collected during measurements: CO_2, air temperature, air humidity, solar panel output current, solar panel output voltage, battery voltage, battery temperature, and battery capacity. The device collected air quality and power parameter data at intervals of 10 minutes, yielding 13,080 rows of data at the end of the measurement period. We used hourly average data for air quality and power parameter estimations. We averaged every six measurements to obtain hourly data, resulting in a new dataset containing 2,180 rows. For missing data imputation, however, we worked on data obtained every 10 minutes of measurements.

We split data into train and test sets by putting 70% of the data in the training set and 30% in the test set. For future parameter estimations, we standardized the features to get a mean of zero and a standard deviation of one, whereas we transformed the features by scaling the features into the range of [0,1] for missing data imputation.

B. Device Design

Fig. 1 shows the hardware interfaces of the low-cost air quality device. A Raspberry Pi (RPi) Pico W is used as the main controller board in this work. This board utilizes RP2040 chip as the microcontroller. This board features a dual-core Cortex-M0+ processor designed with 264kB of SRAM and 2MB of onboard flash memory.

Two power sources for the air quality monitoring device are a solar panel and a 18650 Li-Ion rechargeable battery. A nominal 12V solar panel 20 Watt recharged a Li-Ion battery with nominal voltage and capacity of 3.6V and 3500mAh, respectively. The solar panel

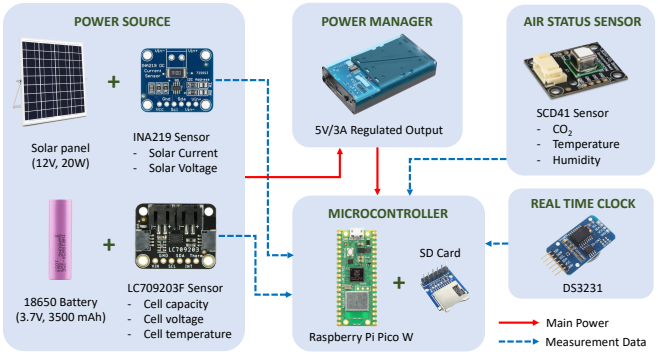

Fig. 1. Module interfaces of the proposed device

is manufactured by Hisunage, China, whereas we used the Li-Ion battery from Samsung SDI. For the solar power manager, we used a product from Waveshare Electronics, China. This solar manager is compatible with general 6V – 24V solar panels and can recharge the 18650 Li-Ion battery. It also provides 5V/3A regulated output, suitable for the RPi pico board supply.

Three sensor modules were utilized in this work: INA219, LC709203F and SCD41. Additionally, a DS3231 real-time clock (RTC) module was also added to enable accurate timekeeping. The INA219 sensor provides abilities to read solar output current and voltage, the LC709203F module determined the battery cell capacity and cell voltage, and the SCD41 sensor module sensed carbon dioxide, temperature, and relative humidity. The collected data were stored on a microSD memory card.

C. TinyML Framework

Our work involved deploying two deep learning models on a single microcontroller: *model predictor* and *model imputer*. The first model is used to run the prediction task, whereas the second model is utilized to impute the missing sensor data. When a sensor fails to collect data in a real-life application, the microcontroller marks this event as missing and perform data imputation before executing a prediction task. During the data collection, we did not find any missing values and trained the model predictor based on the full dataset. In contrast, we deliberately removed measurement values for the model imputer with different levels of missing values. Finally, we compared the prediction results obtained from the full dataset and the dataset with missing data.

We built the deep learning models using TensorFlow (TF) 2.4.0. Developed by Google, TF is an open-source framework primarily designed for deep learning applications [12]. We created, trained and evaluated the models using TF CPU running on a desktop computer. The trained TF models were then converted to TFLite format using TF Lite converter. Furthermore, the TFLite models were quantized using post-training quantization techniques to reduce the deployed model sizes while maintaining their accuracies. The TFLite models were converted to a C byte array and stored in a read-only program memory on the microcontroller. Finally, we run inference on the device using the TFLite for microcontrollers (TFLM) libraries.

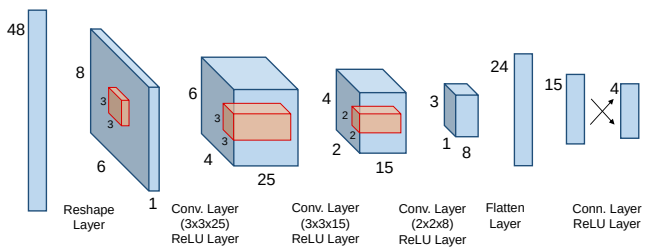

Fig. 2. Model predictor architecture

D. Model Predictor and Model Imputer

For prediction tasks, the model predictor accepts input sets containing eight features: CO_2, air temperature, air humidity, solar panel output current, solar panel output voltage, battery voltage, battery temperature, and battery capacity. The model is designed to predict three air status features (CO_2, air temperature, and air humidity) and one electrical power feature (battery capacity). In this work, we performed a short-term prediction (one hour of future data). The model deployed for the prediction task is shown in Fig. 2

The model predictor accepts input sets consisting of 8 features and 6 hours of historical measurement. Flattening this input, we get 48 as the input size. We reshaped the input sets into three-dimensional data as we utilized two-dimensional convolution layers for the feature extractor and used a fully connected layer with 15 units for the prediction layer. We also utilized rectified linear unit (ReLU) layers as the activation function. For the last layer, no activation is applied.

The model imputer employed in this work is based on autoencoder architecture, and the concept is inspired by image denoising [13]. In this work, the input sets with missing values are considered noisy inputs. In our previous work [7], we implemented this concept by involving spatiotemporal data from neighboring air quality monitoring stations to predict missing values on the target station. In this work, however, we created a simpler, lightweight model suitable for a resource-constraint device. We used local data to train the model without involving spatiotemporal data from other air quality monitoring stations. The denoising autoencoder concept can be shown in Fig. 3.

The encoder function $f_\theta(\cdot)$ and the decoder function $g_\phi(\cdot)$ are parameterized by $\theta = \{\mathbf{W}, \mathbf{b}\}$ and by $\phi = \{\mathbf{W}', \mathbf{b}'\}$, respectively. The $\mathbf{W}$, $\mathbf{b}$ and the $\mathbf{W}'$, $\mathbf{b}'$ represent the weight and bias of the encoder and decoder, respectively. The encoder function is defined as $\mathbf{h} = f_\theta(\mathbf{x})$ and the decoder function as $\mathbf{r} = g_\phi(\mathbf{h})$, where $\mathbf{x}$ is the input, $\mathbf{h}$ is the code representation learning, and $\mathbf{r}$ is the reconstructed input. The perfect condition for model learning is achieved when $g_\phi(f_\theta(\mathbf{x})) = \mathbf{x}$. However, the model cannot learn perfectly but instead try to minimize the error between the actual input and the reconstructed input. For the denoising autoencoder, instead of $\mathbf{x}$, we define $\tilde{\mathbf{x}}$ as the noisy

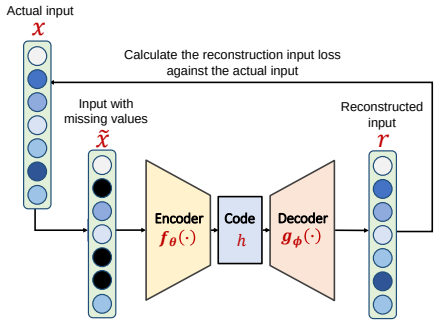

Fig. 3. A denoising convolutional autoencoder workflow

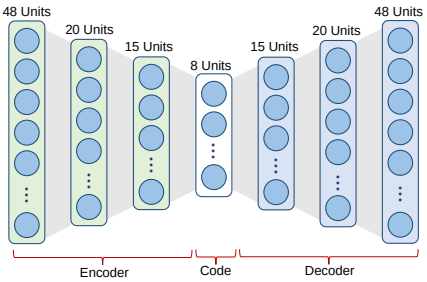

Fig. 4. Model imputer architecture

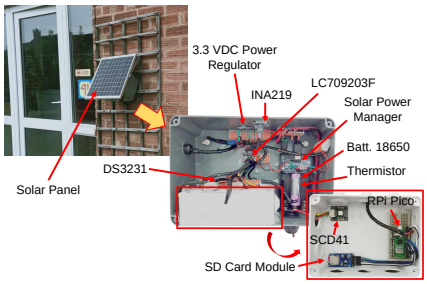

Fig. 5. The low-cost air quality monitoring device

input of $\mathbf{x}$ [14].Then, for each training set $\mathbf{x}^{(i)}$, the parameters θ and ϕ are optimised to minimise the average reconstruction error [13]:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x^{(i)}}, g_\phi(f_\theta(\widetilde{\mathbf{x}}^{(i)}))) \quad (1)$$

where L is the model loss function. The typical loss function is squared error $L(\mathbf{x}, \mathbf{r}) = \|\mathbf{x} - \mathbf{r}\|^2$. Thus, the loss function of the denoising autoencoder is rewritten as:

$$L(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x^{(i)}} - g_\phi(f_\theta(\widetilde{\mathbf{x}}^{(i)})))^2 \quad (2)$$

In this work, the proposed denoising autoencoder consists of several dense layers, as shown in Fig 4. All layers are fully connected, and rectified linear unit (ReLu) layers are used as the activation functions. Similar to the model predictor, the model imputer also accepts input sets consisting of 8 features and 6 hours of measurement. Flattening this input, we get 48 as the input size.

E. Perturbation Procedure

For missing data estimation training and testing, some measurement values were intentionally removed from the input sets, and every deleted value was filled with zero. Following the work conducted by Hadeed et al. [5], four missing rates (i.e., 20%, 40%, 60% and 80% of missing rates) were set in this work. As previously mentioned, the device collected data every 10 minutes, and the model predictor accepted hourly average data. Thus, we assumed that missing data might occur at 10 minutes of measurement level. One sensor can measure more than one parameter. For example, LC709203F measures three parameters: battery capacity, battery voltage and battery temperature. We assumed that these measured parameters would not be obtained if this sensor failed to perform a measurement. Thus, during model training and testing, all parameters measured by the same sensor will have the same missing patterns.

III. RESULTS AND DISCUSSION

A. Device Implementation and Model Performance

The low-cost air quality monitoring device created in this work is shown in Fig. 5. Fig. 5 presents the device with the utilized sensors and other electronic modules. The device was installed in front of the author's house, situated in a suburban area of Coventry city, UK.

A total of 648 test sets were evaluated to predict the averaged 1-hour values of four features: CO_2, air temperature, air humidity, and battery capacity, as reported in Fig. 6. In this evaluation, we used testing data without missing values. The proposed model predictor can estimate each dataset feature with a coefficient of determination above 0.70. The model predicts air humidity most accurately, with an R^2 score of almost 0.9. However, the model seems insensitive to predicting a sharp decline in battery capacity.

While the model predictor accepts hourly mean data, the model imputer works at 10 minutes of measurement level. The air quality monitoring device collects data every 10 minutes, and the model imputer works at this level to fill in any existing missing values. As a result, we can obtain clean sets without missing values. To get hourly data, we averaged every six measurements.

We evaluated the effectiveness of our proposed imputation method on an hourly basis. As shown in Fig. 7, the R^2 score indicates how close the recovered hourly data is to the hourly clean data. The model imputer was trained using training data containing 60% missing values, whereas the missing rate of testing data varied from 20% to 80%. We performed ten experiments with different random seeds to cover various missing patterns. As presented in Fig. 7, the model effectively estimates the testing data with missing rates below 80%. Air temperature, humidity, solar panel voltage and battery temperature achieve the best accuracy with an R^2 score above 0.9, especially at a missing rate of 40%.

B. Post-training Quantization

Converting a trained TF model to the TF Lite format reduces the model size. In this work, we achieved size reductions of 85.4 kilobytes and 102.6 kilobytes for the model predictor and the model imputer, respectively. As we dealt with a tiny, resource-constrained device, we leveraged the post-training options provided by the TensorFlow framework. Due to deep learning architecture, framework version, microcontroller type, and other technical aspects, we only

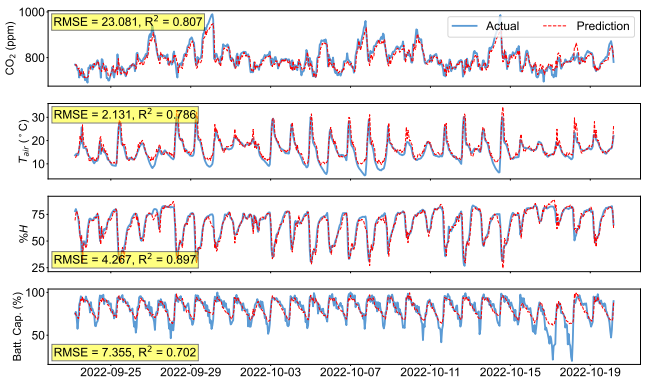

Fig. 6. Performance of model predictor on testing data

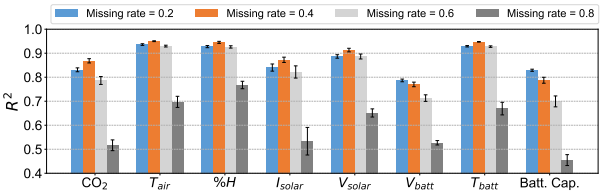

Fig. 7. R^2 scores yielded from different missing rates

TABLE 1. Comparison of different tinyML model sizes

TinyML Model	Model Predictor (bytes)	Model Imputer (bytes)
TF	107,708	126,536
TF Lite	22,280	23,948
TF Lite Quantized	11,648	10,384

TABLE 2. RMSE values of different TF model formats

	Model Predictor			Model Imputer		
Feature	TF	TFL	TFL Q.	TF	TFL	TFL Q.
CO_2(ppm)	23.081	23.081	23.392	26.591	26.591	33.743
T_{air}(°C)	2.131	2.131	2.458	1.176	1.176	1.338
%H	4.267	4.267	4.927	3.447	3.447	4.524
I_{solar}(mA)	-	-	-	29.349	29.349	30.167
V_{solar}(V)	-	-	-	3.203	3.203	3.361
V_{batt}(V)	-	-	-	0.099	0.099	0.101
T_{batt}(°C)	-	-	-	1.02	1.02	1.237
%Batt	7.355	7.355	7.909	12.393	12.393	12.803

successfully performed the integer with float fallback quantization. This quantization technique tries to fully integer quantize the model. However, float operators are still used when the model does not support an integer implementation. By performing this technique, we can further achieve the model size reduction of 10.6 kilobytes and 13.6 kilobytes for the model predictor and the model imputer, respectively. Table 1 reports these model size comparisons.

Table 2 shows $RMSE$ values evaluated from different TF model formats. Converting TF models to TF Lite models (TFL) maintains model accuracies. However, degradations occur after performing quantization (TFL Q.). CO_2 imputation suffers the most, degrading from 26.591 ppm to 33.743 ppm. The $RMSE$ degradations of other features are considered to be less significant.

IV. CONCLUSION

This work involved the development of a low-cost air quality monitoring device with tiny machine learning (tinyML) models to enhance its capabilities. We deployed multiple tinyML models based on 2-D CNN layers and a denoising autoencoder architecture to a single microcontroller and performed parameter prediction and missing feature imputation. The proposed model predictor can estimate the testing data with a coefficient of determination above 0.70, and the model imputer performs better when missing rates are below 80%. With little degradation in model accuracies, the percentage decreases of the quantized versions compared to their original lite model sizes are 47.7% and 56.6% for model predictor and model imputer, respectively.

ACKNOWLEDGMENT

This work was supported in part by Indonesia Endowment Fund for Education (LPDP), Ministry of Finance, Republic of Indonesia under grant number Ref: S-1027/LPDP.4/2019.

REFERENCES

[1] N. Castell et al., “Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?,” Environment International, vol. 99, pp. 293–302, Feb. 2017.

[2] S. Esfahani, P. Rollins, J. P. Specht, M. Cole, and J. W. Gardner, “Smart City Battery Operated IoT Based Indoor Air Quality Monitoring System,” in 2020 IEEE SENSORS, Rotterdam, Netherlands, Oct. 2020.

[3] K. Yadav, V. Arora, M. Kumar, S. N. Tripathi, V. M. Motghare, and K. A. Rajput, “Few-Shot Calibration of Low-Cost Air Pollution (PM2.5) Sensors Using Meta Learning,” IEEE Sensors Letters, vol. 6, no. 5, pp. 1–4, May 2022.

[4] A. C. Rai et al., “End-user perspective of low-cost sensors for outdoor air pollution monitoring,” Science of The Total Environment, vol. 607–608, pp. 691–705, Dec. 2017.

[5] I. N. K. Wardana, J. W. Gardner, and S. A. Fahmy, “Optimising Deep Learning at the Edge for Accurate Hourly Air Quality Prediction,” Sensors, vol. 21, no. 4, p. 1064, Feb. 2021.

[6] Y. Yang, G. Mei, and S. Izzo, “Revealing Influence of Meteorological Conditions on Air Quality Prediction Using Explainable Deep Learning,” IEEE Access, vol. 10, pp. 50755–50773, 2022.

[7] I. N. K. Wardana, J. W. Gardner, and S. A. Fahmy, “Estimation of missing air pollutant data using a spatiotemporal convolutional autoencoder,” Neural Comput & Applic, vol. 34, no. 18, pp. 16129–16154, Sep. 2022.

[8] J. Ma et al., “Transfer learning for long-interval consecutive missing values imputation without external features in air pollution time series,” Advanced Engineering Informatics, vol. 44, p. 101092, Apr. 2020.

[9] M. M. Samsami, N. Shojaee, S. Savar, and M. Yazdi, “Classification of the Air Quality Level based on Analysis of the Sky Images,” in 2019 27th Iranian Conference on Electrical Engineering (ICEE), Apr. 2019, pp. 1492–1497.

[10] M. Z. M. Shamim, “TinyML Model for Classifying Hazardous Volatile Organic Compounds Using Low-Power Embedded Edge Sensors: Perfecting Factory 5.0 Using Edge AI,” IEEE Sensors Letters, vol. 6, no. 9, pp. 1–4, Sep. 2022.

[11] G. M. Iodice, TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter. Packt, 2022. [Online]. Available: https://www.packtpub.com/product/tinyml-cookbook/9781801814973.

[12] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXiv, Mar. 16, 2016. Accessed: Aug. 31, 2022. [Online]. Available: http://arxiv.org/abs/1603.04467

[13] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in Proceedings of the 25th international conference on Machine learning - ICML ’08, Helsinki, Finland, 2008, pp. 1096–1103.

[14] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. 2016. [Online]. Available: http://www.deeplearningbook.org.

[15] S. J. Hadeed, M. K. O’Rourke, J. L. Burgess, R. B. Harris, and R. A. Canales, “Imputation methods for addressing missing data in short-term monitoring of air pollutants,” Science of The Total Environment, vol. 730, p. 139140, Aug. 2020.