# Novel encoder for ambient data compression applied to microcontrollers in agricultural robots

Yueting Wang, Minzan Li, Ronghua Ji, Minjuan Wang, Yao Zhang, Lihua Zheng*

(*Key Laboratory of Smart Agriculture Systems, Ministry of Education, China Agricultural University, Beijing 100083, China*)

**Abstract:** Agricultural robots are flexible to obtain ambient information across large areas of farmland. However, it needs to face two major challenges: data compression and filtering noise. To address these challenges, an encoder for ambient data compression, named Tiny-Encoder, was presented to compress and filter raw ambient information, which can be applied to agricultural robots. Tiny-Encoder is based on the operation of convolutions and pooling, and it has a small number of layers and filters. With the aim of evaluating the performance of Tiny-Encoder, different three types of ambient information (including temperature, humidity, and light) were selected to show the performance of compressing raw data and filtering noise. In the task of compressing raw data, Tiny-Encoder obtained higher accuracy (less than the maximum error of sensors ±0.5 ℃ or ±3.5% RH) and more appropriate size (the largest size is 205 KB) than the other two auto-encoders based convolutional operations with different compressed features (including 20, 60, and 200 features). As for filtering noise, Tiny-Encoder has comparable performance with three conventional filtering approaches (including median filtering, Gaussian filtering, and Savitzky-Golay filtering). With large kernel size (i.e., 5), Tiny-Encoder has the best performance among these four filtering approaches: the coefficients of variation with the large kernel (i.e., 5) were 8.6189% (temperature), 10.2684% (humidity), 57.3576% (light), respectively. Overall, Tiny-Encoder can be used for ambient information compression applied to microcontrollers in agricultural information acquisition robots.

**Keywords:** agricultural information, robot, ambient information, data compression, embedded machine learning methods

**DOI:** 10.25165/j.ijabe.20221504.6911

**Citation:** Wang Y T, Li M Z, Ji R H, Wang M J, Zhang Y, Zheng L H. Novel encoder for ambient data compression applied to microcontrollers in agricultural robots. Int J Agric & Biol Eng, 2022; 15(4): 197–204.

## 1 Introduction

Ambient information is one of the major environmental factors that has a significant impact on many different fields, especially in the agricultural area[1-5]. Adverse environmental conditions severely influence various aspects of plant growth and developmental processes, causing a worldwide reduction in crop yields[6]. To ensure food security, environmental information is needed to timely monitor, transmit, and analyze. The traditional system of ambient information acquisition is often divided into three parts: Wireless Sensor Networks (WSNs), communication base stations, and remote servers[7,8]. WSNs will be deployed in the monitoring area to obtain raw ambient information, which always comprise a large number of sensor nodes. And after transmitting through different communication base stations with different communication networks, environmental information is sent to a remote server for further processing. From the above description, the conventional system of agricultural information acquisition has many potential disadvantages (e.g., fixed, occupation of cropland, power, etc.)[9–12]. For applying to more complex scenarios, more and more agricultural robots are designed and used in practical applications. However, as the amount of collected information increases, more and more data are also needed to transmit. Coding raw data (i.e., source coding, a professional description of the above problems in digital communication systems) is essential to satisfy this requirement: on the one hand, more agricultural information can be represented by smaller transmission cells; on the other hand, the operation of source coding is useful for reducing the effect of noise, which will influence the performance of data analysis[13]. Therefore, designing a good encoder applied to agricultural robots is one of the most important ways to solve the above issues, which is able to compress the raw environmental information and reduce the transmission of redundant information. Moreover, a suitable method of data coding can improve the quality of raw data.

In order to accomplish the aims mentioned, there are many traditional source coding approaches (e.g., Huffman code[14], L-Z code[15], etc.), which bases on a statistical structure depending on the raw data. For better illustration, Huffman code, which is an entropy coding algorithm in information theory and computer science[16], is chosen as a case example to introduce the workflow of conventional source coding. To code different raw data, Huffman code is to represent more frequent data with shorter codes, and less frequent data with longer ones[17]. This approach, however, has one drawback: it only focuses on a certain feature (e.g., the probability and length of code in Huffman code) of raw data and ignores other attributions (e.g., the correlation between adjacent data, etc.). This limits the practicability of conventional approaches. Thanks to the development of machine learning

**Received date:** 2021-07-25 **Accepted date:** 2022-08-11

**Biographies: Yueting Wang**, PhD, research interests: agricultural information, smart agriculture, Email: wyt_study@163.com; **Minzan Li**, PhD, Professor, research interests: agricultural information, smart agriculture, Email: limz@cau.edu.cn; **Ronghua Ji**, PhD, Associate Professor, research interests: agricultural information, Email: jessic1212@cau.edu.cn; **Minjuan Wang**, PhD, Associate Professor, research interest: image processing, Email: minjuan@cau.edu.cn; **Yao Zhang**, PhD, Associate Professor, research interest: agricultural information, Email: zhangyao@cau.edu.cn.

***Corresponding author: Lihua Zheng**, PhD, Professor, research interests: agricultural information. College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China. Email: zhenglh@ cau.edu.cn.

approaches (especially deep learning approaches), more and more data-driven methods are widely used and have achieved a great deal of success in various fields[18-21]. Of these, auto-encoder, deep-learning architectures, can learn the compressed representation from raw datasets and can be used as source coding. Traditional auto-encoder is based on artificial neural network (ANN), but the results obtained from the basic auto-encoder remained significant room for improvement. To improve the performance, there are many new auto-encoder architectures (e.g., long short-term memory (LSTM) encoder[22], stack auto-encoder, sparse auto-encoder, variational auto-encoder, etc.[23]). Although these approaches can overcome the disadvantages of traditional approaches, they are unsuitable for agricultural robots due to a variety of reasons (e.g., model size, large memory, etc.). Therefore, an encoder, which is applied to agricultural robots, should be designed and developed.

Based on the above considerations, an encoder was developed for ambient data compression in this study, which can be applied to agricultural information acquisition robots. This encoder is named Tiny-Encoder and is based on convolutional and pooling operations. To evaluate the performance of compression, Tiny-Encoder was used to compress three kinds of ambient information (including temperature, humidity, and light) and it performed better than the other two auto-encoders based convolutional operations with different compressed features (including 20, 60, and 200 features). Besides, Tiny-Encoder was compared with three traditional filtering methods (including median filtering, Gaussian filtering, and Savitzky-Golay filtering) for removing noise. The results demonstrate that Tiny-Encoder has comparable performance with three conventional filtering approaches. Overall, Tiny-Encoder can be used for ambient information compression applied to agricultural information acquisition robots.

## 2 Materials and methods

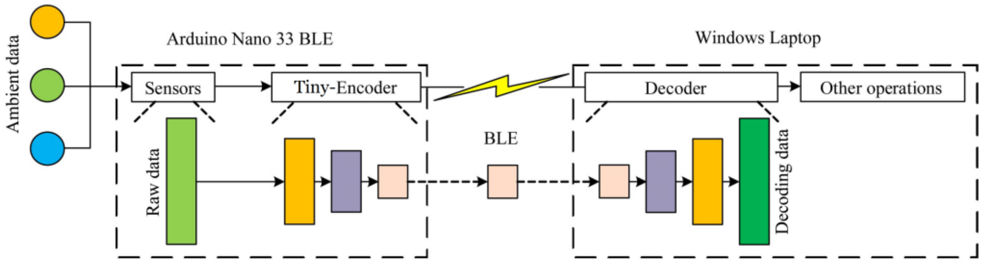
### 2.1 Platform for collecting ambient information and processing

Arduino Nano 33 Bluetooth Low Energy (BLE) was selected as an example device to collect environmental information in this work, and it is deployed Tiny-Encoder. Arduino Nano 33 BLE Sense embeds a 32-bit ARM Cortex-M4 CPU (nRF52840) running at 64 MHz, which has 1 MB of program memory. Besides, Arduino Nano 33 BLE Sense comes with a series of embedded sensors (including a temperature and humidity sensor (HTS221), an ambient light sensor (APDS-9960), etc.), and it can be used to obtain temperature (±0.5 ℃, 15 to +40 ℃), humidity (±3.5% RH, 20 to +80% RH), and light (0 to 4097 lx). It should be specially explained that the monitored ranges of light intensity are adjustable, which is based on the integration time (set as 10 ms in this experiment) and count register size (set as 16 bits). In this work, experimental samples were collected between 5:00 and 8:00 am on May 3, 2021, and detailed characteristics are presented in Table 1.

**Table 1 Description of raw ambient data**

| Total samples | Temperature/℃ | | | Humidity/(% RH) | | | Light/lx | | |
|---|---|---|---|---|---|---|---|---|---|
| | Min | Max | Mean | Min | Max | Mean | Min | Max | Mean |
| 10 500 | 15.9 | 22.19 | 17.14 | 27.96 | 43.91 | 37.78 | 20 | 4097 | 2623.95 |

Tiny-Encoder is implemented on the Python platform using the TensorFlow Lite library, and the part deployed in Arduino Nano 33 BLE was written in C++, which is transformed by a Linux command, xxd. The process was performed on a Windows laptop (Windows 10) with 16 GB of RAM, and an Nvidia Geforce GTX 1650 graphics card with 4 GB of RAM. The schematic diagram of data streams for this study is shown in Figure 1.



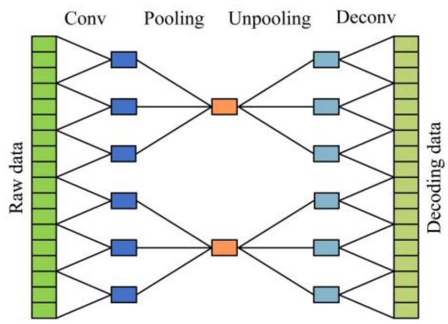

Note: BLE: Bluetooth Low Energy.

Figure 1 Schematic diagram of data streams

### 2.2 Auto-encoder based on convolutional and pooling operations

The aim of the auto-encoder is to select the characteristic features of raw ambient information, which can revert to the original data. Typically, the auto-encoder contains two parts: encoder and decoder. The characteristic features of raw data were selected in the encoding phase, and the original data were recovered in the decoding phase. The ambient data were associated with acquisition time, and commonly, the mutation does not occur. Therefore, the relationships between the ambient data should be considered. In response to this demand, convolutional and pooling operations have a better performance: convolutional and pooling operations can be used to capture and compress the characteristic features (i.e., encoder)[24,25]. Deconvolutional and un-pooling operations can be thought of as an inverse operation to convolutional and max-pooling operations (i.e., decoder)[26].

As shown in Figure 2, an example of two layers with the convolutional/deconvolutional layer having a filter (the size of the filter is three and strid is also three). The pooling size of the max pooling/un-pooling layer is three. The input of the auto-encoder based on convolutional operations is raw environmental data (i.e., green rectangles) and the output is decoded data (i.e., light green rectangles). One neuron after the pooling layer (i.e., orange rectangle) covers 9 original ambient data, respectively.




Note: Conv: Convolution; Deconv: Deconvolution.

Figure 2 Diagram of auto-encoder based on convolutional and pooling operations

The key point for the auto-encoder is to determine the number of hidden nodes, which are influenced by the number from one-channel (Figure 2) and channel size. Suppose the kernel size of the convolutional layer is $k$, the stride size is $s$ and the padding size is $p$. When the numbers of features are $i$ after the previous convolutional layer, the numbers of features in the output layer ($o$) from one channel are able to be computed by Equation (1). Through the max-pooling layer, in which pooling size is Pooling_size, the number of features from one-channel ($o'$) are calculated by Equation (2). The final number of hidden layers for auto-encoder is the channel size of the last layer multiplied by the number from one-channel.

$$o = \frac{i + 2p - k}{s} + 1 \tag{1}$$

$$o' = \frac{o}{Pooling\_size} \tag{2}$$

Considering application to microcontrollers, simple structure is an important factor other than accuracy. Therefore, the auto-encoder was designed as shown in Figure 3. This auto-encoder only has 4 layers (2 convolutional layers (labeled as Conv) and 2 pooling layers) in the structure of the encoder (i.e., Tiny-Encoder), 2 deconvolutional layers, and 2 un-pooling layers in the decoder. The input of the auto-encoder is raw ambient data and the output is decoded data. The length of to-be-transmitted characteristics can be controlled by the parameters of convolutional and pooling layers.

**2.3 Design of experiments**

2.3.1 Performance of compressing raw ambient data with different models

The performance of compressing raw ambient data is directly affected by the structure of the auto-encoder. To evaluate the effect of different architectures, two other auto-encoders were built based on the convolutional and pooling operation (shown in Figure 4). The major difference between these models lies in the numbers of layers and filters, which will impact the number of characteristic features and the size of the models.

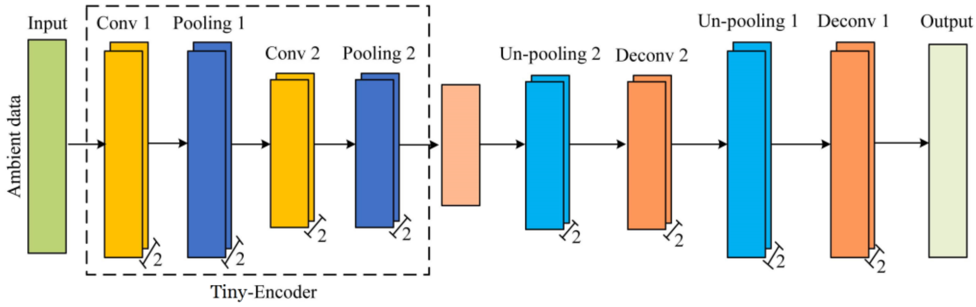
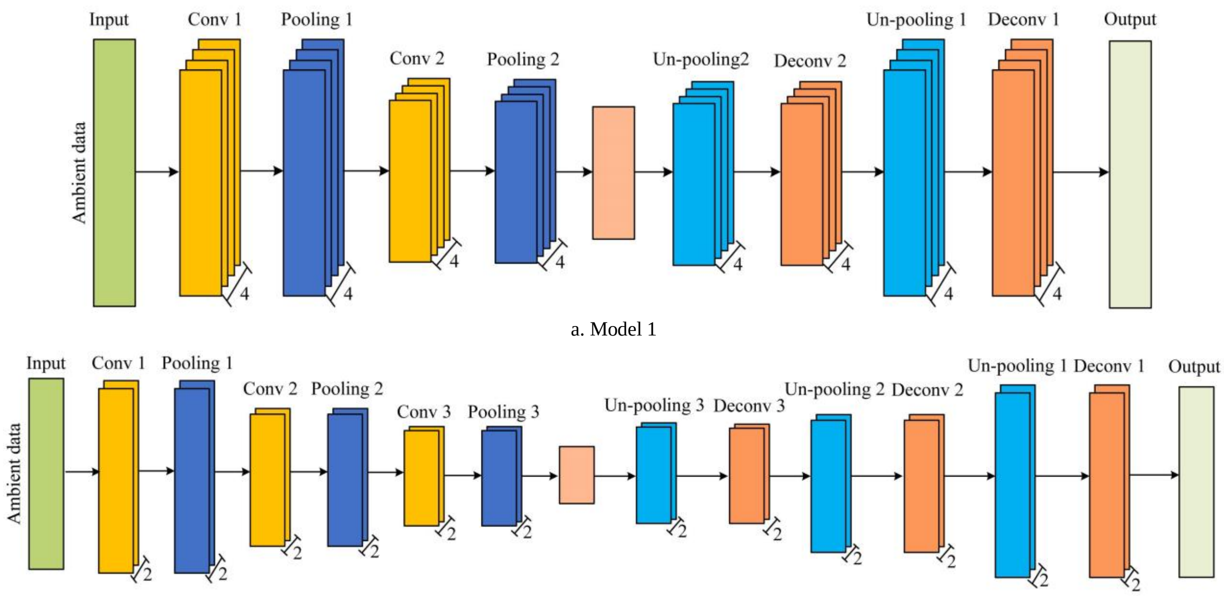



Figure 3 Diagram of auto-encoder that contains Tiny-Encoder




a. Model 1

b. Model 2

Figure 4 Architectures of two autoencoders based on convolutional and pooling operations

The input of each model is three different types of ambient data (including temperature, humidity, and light), which are measured by one Arduino Nano 33 BLE Sense inside the chamber. The convolutional/deconvolutional layers are labeled as Conv/Deconv and max-pooling/un-pooling layers are labeled as Pooling/ Un-pooling. Due to the memory size of Arduino Nano 33 BLE Sense, each model only has fewer numbers of layers and each layer has fewer numbers filters. In this way, designed auto-encoders will be thinner and more compact. In this work, zero-padding is used to retain the edge information, and the rectified linear unit (ReLU) is selected as the activation function for the convolutional layers in each model. Besides, Adam optimizer was selected to search the local minimum of the objective function. Mean squared error (MSE) is adopted as the loss function, which is presented in Equation (3).

$$Loss = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_l)^2}{n} \tag{3}$$

where, $y_i$ and $\hat{y}_i$ [[are measured values and predicted values, respectively; $n$ is the number of samples in the training set; $i$ is the *i*-th sample.

2.3.2 Performance of ambient data transfer with different filter processing

Due to the effect of the communication environment and sensor status (e.g., electromagnetic wave, the temperature of the device, etc.), there is usually a lot of noise in the raw ambient data. Filtering out noise is one of the important goals of source encoding. With traditional approaches, the small size of raw data was firstly selected, and then stable information was picked out by different operations (e.g., mean) according to these samples mentioned. Other new data can be generated by repeating the process above. Different from conventional methods, Tiny-Encoder removes the noise during the operations of convolution and pooling. To evaluate the performance of Tiny-Encoder with the task of moving noise, three common algorithms (including median filtering[27], Gaussian filtering[28], and Savitzky-Golay (SG) filtering[29]) were selected to compare with Tiny-Encoder, and coefficient of variation (CV, Equation (4)) to measure the performance of different filters, a smaller CV indicates a better result.

$$CV = \frac{\delta}{\mu} \times 100\% \tag{4}$$

where, $\delta$ is the standard deviation; $\mu$ is the mean of target variables.

## 3 Results and discussion

### 3.1 Performance of data compressing with different models

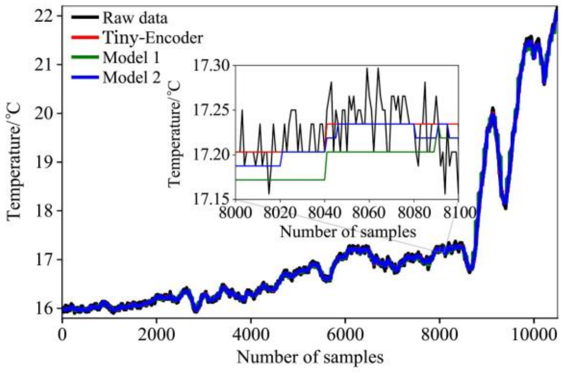
In the first experiment, 10 500 samples are divided into 35 groups (300 samples/group, which were collected in 5 min) as training samples. To analyze the effect of feature numbers, 300 samples in each group are compressed to 200, 60, and 20 characteristic features according to the pooling operation with different pooling sizes (Table 2), respectively. The results are shown in Figure 5. Note that, the kernel size of different convolutional layers was 3, the stride was 1, and zero-padding was set to preserve the raw dimensions. Moreover, the batch size was 256, and the size of the epoch was 15 000.

**Table 2 Hyperparameters set in different models**

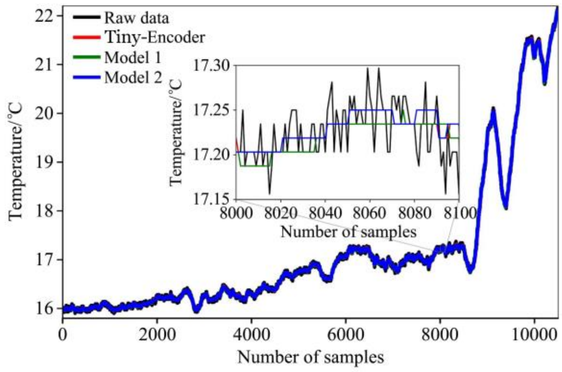
| Model | Features | Pooling 1 | Pooling 2 | Pooling 3 |
|---|---|---|---|---|
| Tiny-Encoder | 200 | 3 | 1 | -- |
| | 60 | 5 | 2 | -- |
| | 20 | 6 | 5 | -- |
| Model 1 | 200 | 3 | 2 | -- |
| | 60 | 5 | 4 | -- |
| | 20 | 10 | 6 | -- |
| Model 2 | 200 | 3 | 1 | 1 |
| | 60 | 5 | 2 | 1 |
| | 20 | 5 | 3 | 2 |

As it can be seen from Figure 5, all these three models are able to maintain the trends of raw data. With the number of layers increasing, decoding data can retain more information than the original data (e.g., Model 2). This is because models with more layers can capture more different features of raw ambient data. It can also increase the number of filters in each layer to maintain more details of raw data, but the performance is weaker than the model with more layers. Differences between raw ambient data and decoding data among the three models are less than the maximum error of sensors (±0.5 ℃ or ±3.5% RH).

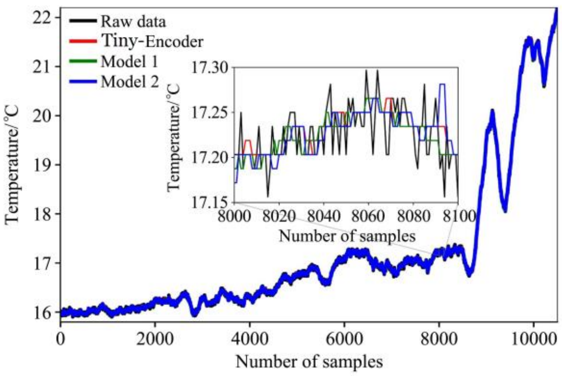
As noted earlier, the performance of recovery data is affected directly by the number of features (instead of model architecture), which are selected by max-pooling layers. The numbers of layers and filters in convolutional layers only affect the content of features, but pooling size will directly affect the number of features. More filters (e.g., Model 1) or layers (e.g., Model 2) are more sensitive to details of raw ambient data, this is because these models capture more features for each neuron in convolutional layers[30,31]. To transmit the same numbers of data, Model 1 and Model 2 need to pick out fewer features in pooling layers. In other words, they will discard more information. Tiny-Encoder takes this into account and has fewer layers and each layer has fewer filters.



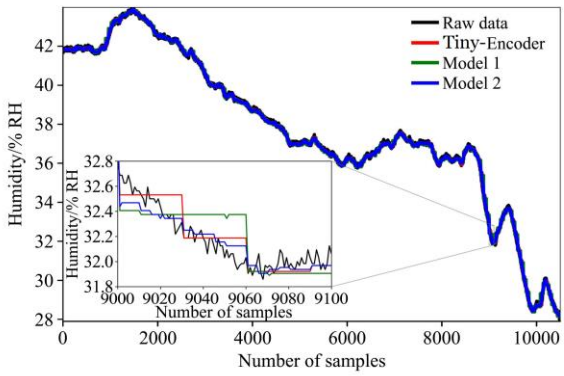

a. Decoding temperature with 20 features



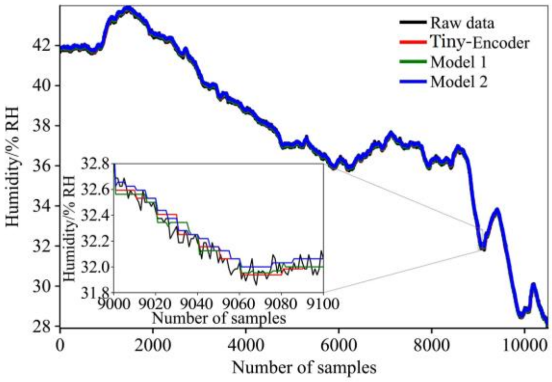

b. Decoding temperature with 60 features



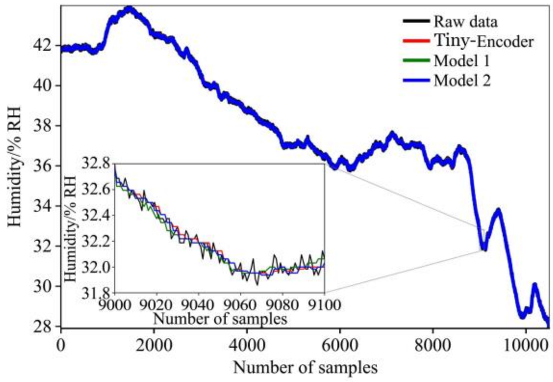

c. Decoding temperature with 200 features



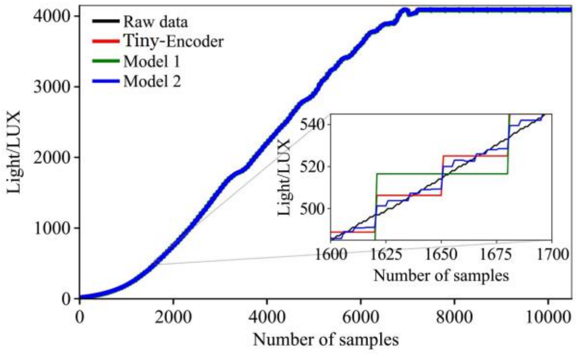

d. Decoding humidity with 20 features



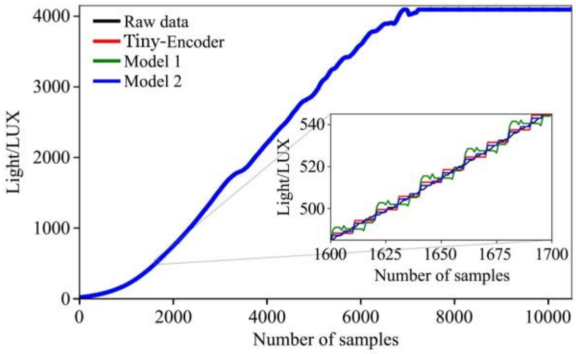

e. Decoding humidity with 60 features



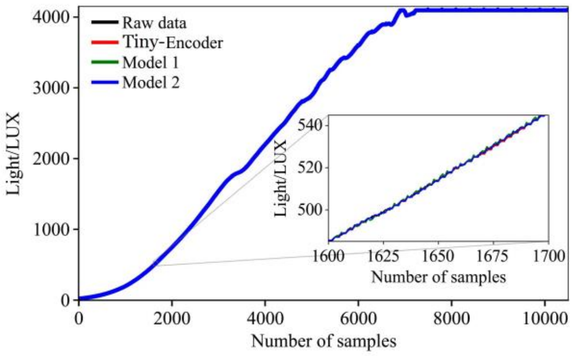

f. Decoding humidity with 200 features




g. Decoding light with 20 features




h. Decoding light with 60 features




i. Decoding light with 200 features

Figure 5 Results of decoding ambient data with different models

Statistical characteristics of the ambient data are more important than instantaneous data in the agriculture domain. Therefore, the statistical characteristics were compared of decoding data with different models, and results are shown in Table 3. As can be seen, all these three models are able to obtain good performance when the data variance is small. Besides, Tiny-Encoder can obtain the best results with the variable of light, which has the largest data variance in the three types of ambient data. The results obtained in the experiments provide enough evidence of the usefulness of Tiny-Encoder.

**Table 3 Statistical characteristics of decoding data with different models.**

| Data source | Features | Temperature/ ℃ | | | Humidity/(% RH) | | | Light/lx | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Max | Mean | Min | Max | Mean | Min | Max | Mean |
| Raw data | -- | 15.90 | 22.19 | 17.14 | 27.96 | 43.91 | 37.78 | 20 | 4097 | 2623.95 |
| Tiny-Encoder | 200 | 15.93 | 22.15 | 17.14 | 28.01 | 43.86 | 37.77 | 19.68 | 4095.95 | 2623.05 |
| | 60 | 15.95 | 22.11 | 17.14 | 28.04 | 43.84 | 37.77 | 19.12 | 4100.43 | 2625.22 |
| | 20 | 15.96 | 22.04 | 17.14 | 28.18 | 43.83 | 37.77 | 14.77 | 4099.59 | 2623.65 |
| Model 1 | 200 | 15.93 | 22.16 | 17.14 | 27.99 | 43.84 | 37.77 | 17.42 | 4096.90 | 2623.63 |
| | 60 | 15.95 | 22.10 | 17.14 | 28.04 | 43.84 | 37.78 | 14.14 | 4098.16 | 2623.61 |
| | 20 | 15.94 | 21.96 | 17.14 | 28.14 | 43.80 | 37.77 | 7.34 | 4098.06 | 2619.34 |
| Model 2 | 200 | 15.93 | 22.18 | 17.14 | 28.02 | 43.92 | 37.77 | 20.53 | 4108.93 | 2623.68 |
| | 60 | 15.95 | 22.11 | 17.14 | 28.11 | 43.93 | 37.84 | 18.62 | 4099.17 | 2623.56 |
| | 20 | 15.95 | 22.07 | 17.14 | 28.17 | 43.90 | 37.77 | 13.57 | 4099.25 | 2622.77 |

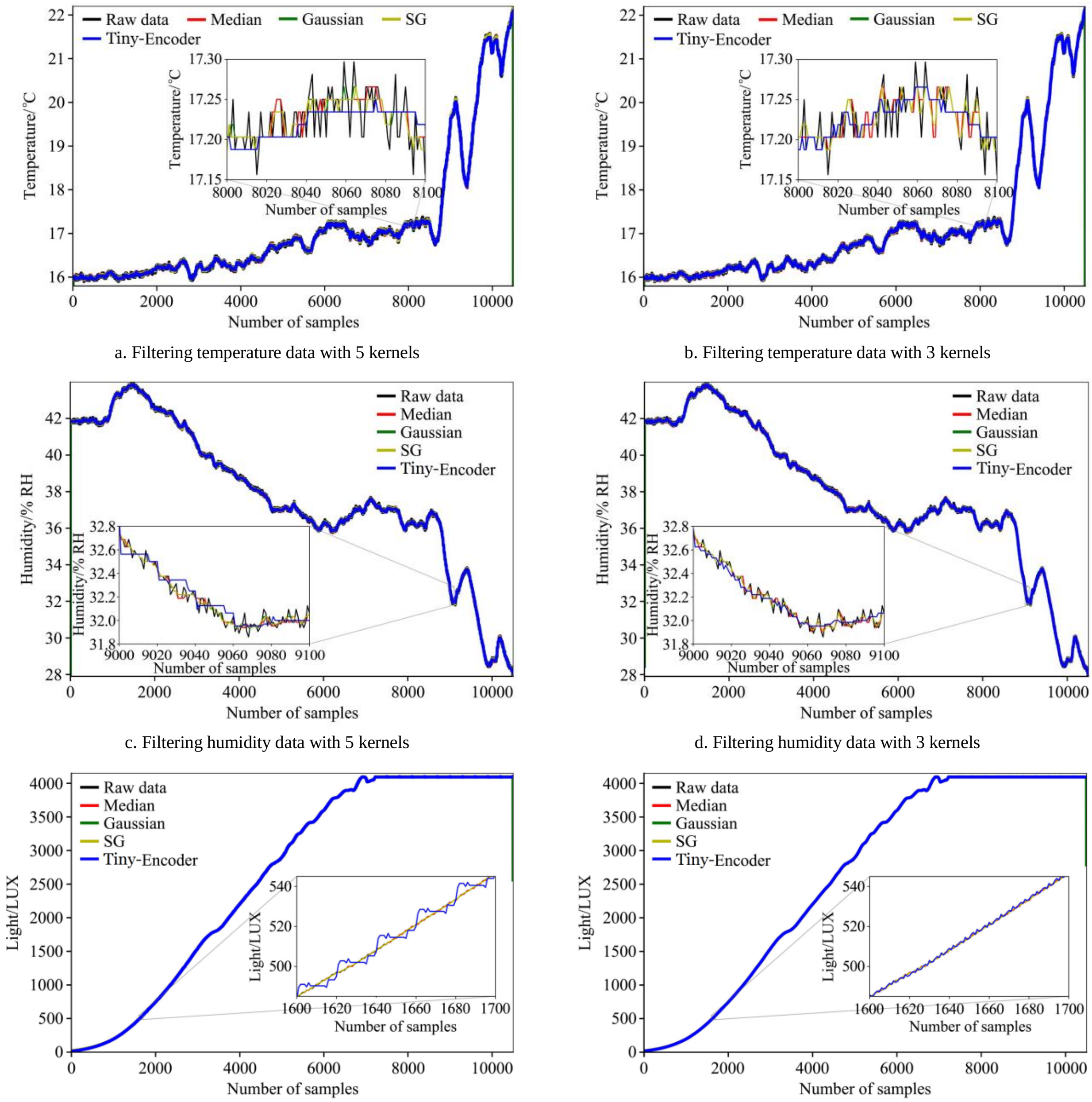
In order to apply to Arduino Nano 33 BLE Sense, the other critical point is the size of models (shown in Table 4). Different ambient information does not affect the size of models, so only one value was shown with different ambient information. According to Table 4, Model 1, which has four filters in different convolutional layers and has the least number of features in each layer, obtained the smallest size of models with different features, Tiny-Encoder is the second smallest, and Model 2 has the largest size of the model. This implies that the number of features in each layer affected more model size and the model size will increase with the increasing number of layers.

### 3.2 Performance of data filtering with different methods

The effect of filters is mainly affected by the small range scanning, which can be set as an artificial upper limit. In order to compare the performance of different filtering, 3 and 5 were selected as the kernel sizes of different filters (including median filters, Gaussian filters, SG filters, and Tiny-Encoder) to filter the raw ambient data, and the results are shown in Figure 6.

**Table 4 Statistical characteristics of decoding data with different models**

| Model | Tiny-Encoder | | | Model 1 | | | Model 2 | | |
|---|---|---|---|---|---|---|---|---|---|
| Features | 20 | 60 | 200 | 20 | 60 | 200 | 20 | 60 | 200 |
| Size/KB | 107 | 127 | 205 | 92 | 119 | 173 | 152 | 173 | 298 |




a. Filtering temperature data with 5 kernels
b. Filtering temperature data with 3 kernels
c. Filtering humidity data with 5 kernels
d. Filtering humidity data with 3 kernels
e. Filtering light data with 5 kernels
f. Filtering light data with 3 kernels

Figure 6 Performance of different filters with different kernels

According to Figure 5, with a smaller kernel, different filters can retain more detail and the filtered data are closer to the original data. With a larger kernel, median filters and Tiny-Encoder have more flat-top and the other filters can retain more details of raw data. The reason is that Tiny-Encoder and median filter the noise by sampling (median or max), and Gaussian filter and SG filtering filter the noise by fitting the data. In order to further examine the effect of different filtering, we compared the CV of different filtering (Table 5). Note that the results are decimal and up to 4 digits.

According to Table 5, it is known that Tiny-Encoder has a little different from the other filtering, and it has strong stability with different kernel sizes. Even in many cases (e.g., filtering temperature and light), Tiny-Encoder has the optimal results. Medina filter with smaller kernel size performs better than lager. This is because ambient information will not change suddenly, data in a small range is stable. Gaussian filter is based on an assumption that ambient data is accorded with Gaussian distribution. However, it is not obvious in real applications. As for the SG filter, it is the most stable filter, but the performance is worse than Tiny-Encoder.

**Table 5 Performance of filters with different kernels**

| Kernel size | Methods | Coefficient of variation | | |
|---|---|---|---|---|
| | | Temperature | Humidity | Light |
| 5 | Median | 8.6201% | 10.2684% | 57.3599% |
| | Gaussian | 8.6251% | 10.2867% | 57.3601% |
| | Savitzky-Golay | 8.6202% | 10.2692% | 57.3599% |
| | Tiny-Encoder | 8.6189% | 10.2684% | 57.3576% |
| 3 | Median | 8.6194% | 10.2689% | 57.3599% |
| | Gaussian | 8.6231% | 10.2799% | 57.3601% |
| | Savitzky-Golay | 8.6204% | 10.2693% | 57.3599% |
| | Tiny-Encoder | 8.6199% | 10.2690% | 57.3591% |

## 4 Conclusions

Agricultural robots are flexible to obtain ambient information across large areas of farmland. But it needs to face two major challenges: data compression and filtering noise. To address these challenges, an encoder, named Tiny-Encoder, is presented and it can be applied to agricultural robots. The following conclusions are accordingly achieved.

1) Tiny-Encoder can compress ambient information (e.g., temperature, humidity, and light). Compared with the other two auto-encoders based convolutional operations, Tiny-Encoder obtained higher accuracy (less than the maximum error of sensors ±0.5 ℃ or ±3.5% RH) and more appropriate size (the largest size is 205 KB) with different compressed features (including 20, 60, and 200 features).

2) Tiny-Encoder has comparable performance with conventional filtering approaches (e.g., median filtering, Gaussian filtering, and Savitzky-Golay filtering). Tiny-Encoder has the best performance with the larger kernel: the values of coefficient of variation with the large kernel (i.e., 5) are 8.6189% (temperature), 10.2684% (humidity), 57.3576% (light), respectively.

Tiny-Encoder is a new approach for ambient data compression, which can be applied into agricultural robots. We only considered a simple application of ambient data compression in this work, and there are many problems (i.e., missing data, concurrent, and etc.) may be encountered in the practical application. Moreover, with the development of hardware and software, more and more deep learning algorithms can be applied to embedded devices, and these works will be the focus of our future studies.

## Acknowledgements

This work was financially supported by the National Key Research and Development Program (Grant No. 2019YFE0125500) and the Chinese University Scientific Fund (Grant No. 2021TC111).

## [References]

[1] Quint M, Delker C, Franklin K A, Wigge P H, Halliday K J, van Zanten M. Molecular and genetic control of plant thermomorphogenesis. Nature Plants, 2016; 2: 15190. doi: 10.1038/nplants.2015.190.

[2] Liu H J, Gao Z Z, Zhang L W, Liu Y. Stomatal conductivity, canopy temperature and evapotranspiration of maize (*Zea mays* L.) to water stress in Northeast China. Int J Agric & Biol Eng, 2021; 14(2): 112–119.

[3] Sun Y C, Bao E C, Zhu C M, Yan L L, Cao Y F, Zhang X H, et al. Effects of window opening style on inside environment of solar greenhouse based on CFD simulation. Int J Agric & Biol Eng, 2020; 13(6): 53–59.

[4] Shrestha S, Sharma S. Assessment of climate change impact on high flows in a watershed characterized by flood regulating reservoirs. Int J Agric & Biol Eng, 2021; 14(1): 178–191.

[5] Li H Y, Wang Y H, Xue J, Xie R Z, Wang K R Zhao Rl, et al. Allocation of maize varieties according to temperature for use in mechanical kernel harvesting in Ningxia, China. Int J Agric & Biol Eng, 2021; 14(1): 20–28.

[6] Abade A, Ferreira P A, de Barros Vidal F. Plant diseases recognition on images using convolutional neural networks: A systematic review. Computers and Electronics in Agriculture, 2021; 185: 106125.

[7] Xie C J, Zhang D X, Yang L, Cui T, Zhong X J, Li Y H, et al. Remote monitoring system for maize seeding parameters based on android and wireless communication. Int J Agric & Biol Eng, 2020; 13(6): 159–165.

[8] Wang Y T, Li M Z, Ji R H, Wang M J, Zhang Y, et al. A convolutional operation-based online computation offloading approach in wireless powered multi-access edge computing networks. Computers and Electronics in Agriculture, 2022; 197: 106967. doi: 10.1016/ j.compag.2022.106967.

[9] Hosny K M, Khashaba M M, Khedr W I, Amer F A. New vertical handover prediction schemes for LTE-WLAN heterogeneous networks. Plos One, 2019; 14(4): 0215334. doi: 10.1371/journal.pone.0215334.

[10] Guo S T, Xiao B, Yang Y Y, Yang Y. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE INFOCOM 2016, 2016; pp.1-9. doi: 10.1109/INFOCOM.2016.7524497.

[11] Tao W, Zhao L, Wang G W, Liang R B. Review of the internet of things communication technologies in smart agriculture and challenges. Computers and Electronics in Agriculture, 2021; 189: 106352. doi: 10.1016/j.compag.2021.106352.

[12] Perera T D P, Jayakody D N K, Chatzinotas S, Sharma V. Wireless information and power transfer: Issues, advances, and challenges. In: 2017 IEEE 86th. Technol. Conf. 2017-Septe (2018) 1–7. https://doi.org/Vehicular Technology Conference (VTV-Fall), 2017; pp.1-7. doi: 10.1109/VTCFall.2017.8288396.

[13] ARandelhoff A, Lacour L, Marec C, Leymarie E, Lagunas J, Xing X G, Darnis G, et al. Arctic mid-winter phytoplankton growth revealed by autonomous profilers. Science Advances, 2020; 6(39): abc2678. doi: 10.1126/sciadv.abc2678.

[14] Huffman D A. A method for the construction of minimum-redundancy codes. Resonance, 2006; 11: 91–99.

[15] Lee H, Kim K, Choi O, Yeh H. Code based compression method for effectively using the memory of embedded devices. International Journal of Communication Systems, 2013; 26(6): 747–756.

[16] Erguzen A, Erdal E. An efficient middle layer platform for medical imaging archives. Journal of Healthcare Engineering, 2018; Article ID 3984061. doi: 10.1155/2018/3984061.

[17] Chen C A, Chen S L, Huang H Y, Luo C H. An efficient micro control unit with a reconfigurable filter design for wireless body sensor networks (WBSNs). Sensors, 2012; 12(21): 16211-16227.

[18] Wang Y T, Li M Z, Ji R H, Wang M J, Zhang Y, Zheng L H. Mark-Spectra : A convolutional neural network for quantitative spectral analysis overcoming spatial relationships. Computers and Electronics in Agriculture, 2022; 192: 106624. doi: 10.1016/j.compag.2021.106624.

[19] Li Z B, Guo R H, Li M, Chen Y R, Li G Y. A review of computer vision technologies for plant phenotyping. Computers and. Electronics in Agriculture, 2020; 176: 105672. doi: 10.1016/j.compag.2020.105672.

[20] Jin W, Lim S, Woo S, Park C, Kim D. Decision-making of IoT device operation based on intelligent-task offloading for improving environmental optimization. Complex & Intelligent Systems, 2022. doi: 10.1007/ s40747-022-00659-z.

[21] Porumb M, Stranges S, Pescape A, Pecchia L. Precision medicine and artificial intelligence: A pilot study on deep learning for hypoglycemic events detection based on ECG. Scientific Reports, 2020; 10: 170. doi: 10.1038/s41598-019-56927-5.

[22] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997; 9(8): 1735–1780.

[23] Ai D M, Wang Y Du, Li X X, Pan H F. Colorectal cancer prediction based on weighted gene co-expression network analysis and variational auto-encoder. Biomolecules, 2020; 10(9): 1207. doi: 10.3390/ biom10091207.

[24] Yu Y, Long J, Cai Z P. Network intrusion detection through stacking dilated convolutional autoencoders. Security and Communication Networks, 2017; Article ID 4184196. doi: 10.1155/2017/4184196.

[25] Moran A, Alonso S, Perez D, Prada M A, Fuertes J J, Dominguez M. Feature extraction from building submetering networks using deep learning. Sensors, 2020; 20(13): 3665. doi: 10.3390/s20133665.

[26] Shelhamer E, Long J, Darrel T. Fully convolutional networks for semantic segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017; 39(4): 640–651.

[27] Zhou Y T, Li S H, Zhang D, Chen Y K. Seismic noise attenuation using an online subspace tracking algorithm. Geophysical Journal International, 2018; 212 (2): 1072–1097.

[28] You Z Z, Balbastre Y, Bouvier C, Herard A-S, Gipchtein P, Hantraye P, et al. Automated individualization of size-varying and touching neurons in macaque cerebral microscopic images. Frontiers in Neuroanatomy, 2019; 13: 98. doi: 10.3389/fnana.2019.00098.

[29] Gupta S, Huang C H, Singh G P, Park B S, Chua N H, Ram R J. Portable Raman leaf-clip sensor for rapid detection of plant stress. Scientific Reports, 2020; 10: 20206. doi: 10.1038/s41598-020-76485-5.

[30] Wang Y T, Li M Z, Ji R H, Wang M, Zhang Y, Zheng L H. Construction of complex features for predicting soil total nitrogen content based on convolution operations. Soil and Tillage Research, 2021; 213: 105109. doi: 10.1016/j.still.2021.105109.

[31] Szegedy C, Liu W, Jia Y Q, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recogniction, 2015; pp. 1–9. doi: 10.1109/ CVPR.2015.7298594.