iBUG: AI Enabled IoT Sensing Platform for Real-time Environmental Monitoring

Md Faishal Yousuf
Electrical and Computer Engineering
University of New Hampshire
Durham, USA
mdfaishal.yousuf@unh.edu

Talha Siddique
Electrical and Computer Engineering
University of New Hampshire
Durham, USA
talha.siddique@unh.edu

MD Shaad Mahmud
Electrical and Computer Engineering
University of New Hampshire
Durham, USA
mdshaad.mahmud@unh.edu

***Abstract*—Microcontroller-based smart devices and sensing systems have exploded in popularity in recent years, owing to the growing adoption of Internet of Things (IoT) platforms. Due to the widespread deployment of environmental sensing technology and digitization, data creation has surged to never-before-seen levels. However, it is computationally expensive to transport all these data to the cloud systems for decision making specially for battery operated sensor nodes. As a result, edge AI has arisen as a viable alternative to traditional AI, capable of making simple decisions without the assistance of any cloud system, making it more energy efficient and reducing the requirement for continuous data transfer. A single-board IoT platform capable of performing slimmer version of machine learning with multiple wireless protocols have become increasingly popular for this purpose. In this paper we demonstrate an energy efficient IoT platform (iBUG) that already has numerous built-in sensors. We also present an algorithm utilizing TensorFlow lite to accurately predict CO_2 using only gas resistance.**

***Index Terms*—Real-time, Machine Learning, Sensor, IoT, TinyML, Edge AI**

I. Introduction

The IoT platform has played a crucial role across a variety of areas, particularly in environmental sensing due to massive connectivity, by generating a tremendous amount of granular data and new revenue streams [1]. This will become evermore demanding for wireless communication with edge-cloud computing since the edge computer is the close proximity of the IoT devices, though short-long range communications (e.g., WiFi, Bluetooth, LoRa). Long Range Wide Area Networks (LoRaWAN) have recently received significant traction as a way to collect data via WSN. LoRa is an open-source Long-Range wireless technology that allows low-power devices to interact with Internet-connected applications across long distances [2]. It is a common consensus that the next generation wireless sensor network should be designed to perform efficient and reliable computing with multiple wireless protocols.

Recently, a significant amount of effort has gone into improving embedded technologies for usage in resource-constrained contexts. For example, edge computing, in which data is processed in part at the network edge rather than wholly in the cloud, has been fueled by the development of the IoT and rich cloud services. An edge computing capable platform could improve the latency and battery life on mobile devices, bandwidth costs, security, and privacy. Since cloud computing has more computational capacity than the devices at the network edge, moving all computing services to the cloud server has been a cost-effective approach to process data [3]. However, while data processing speeds have increased significantly, network bandwidth that transports data to and from the cloud server has not. As a result, the network becomes a bottleneck for cloud computing, as edge devices generate more data. Data processing at the network edge would result in faster response times, more efficient processing, reduce power consumption and less network load.

This paper presents iBUG, a low-powered edge node coupled with a vast range of sensors and capable of embedded machine learning and wireless data transfer though multiple wireless protocols (WiFi, BLE, LoRa). We also developed a slimmer version of the TensorFlow algorithm to accurately predict CO_2 content in ambient air using only gas resistance. This article is organized as follows. In the following section, we present the current state-of-the-art studies on AI enabled sensing board and motivation behind the proposed research work. Then the proposed hardware design, where we discussed the unique features and on-board sensors of the iBUG platform. After this section, a developed algorithm using an embedded machine learning algorithm was discussed, where we also optimized data collection and processing methods. Finally, results section and open research issue to conclude.

II. Background and Motivation

A. Internet of Things (IoT)

The IoT is one of the most widely discussed topics in scientific research nowadays. The core premise of IoT is to collect data from the real world using billions of intelligent devices and transfer it to internet repositories for further processing to create novel services. Smart farming, autonomous vehicles, pollution monitoring, industrial automation, smart city, and other scenarios are only a few examples. The data produced at the edge node of an IoT network transmitted using

This work was supported by National Science Foundation SitS Award: **1920965**

978-1-6654-6816-9/22/$31.00 ©2022 IEEE

TABLE I: Commercially available microcontrollers

SBC	Architecture	Price
Raspberry Pi Zero 2 W	ARM, 32-bit	$15
Arduino Nano 33 BLE Sense	nRF52840	33.40
Apollo3 Blue	ARM	16.50
Adafruit EdgeBadge	ARM	35.95
Pine A64-LTS	ARM, 64-bit	$32
PocketBeagle	ARM, 32-bit	$35
Raspberry Pi 4 Model B	ARM, 32-bit	From $45
Odroid-C4	ARM, 64-bit	$54
Rock Pi 4 Model C	ARM, 64-bit	$59
Pine64 ROCKPro 64	ARM, 64-bit	$59/79
Raspberry Pi Pico	ARM, 32-bit	$4
BBC Micro:Bit V2	ARM, 32-bit	$16
Arduino Mega 2560	RISC, 8-bit	$40

various wireless technologies like Bluetooth, Wi-Fi, 3G, 4G etc. However, due to power and range limitations, different technologies have been designed for IoT applications (e.g, LoRA, Sigfox, NB-IoT, LTE-M). Among them, LoRa is the most popular, and there are many commercially available LoRa nodes in the market. For example, Park et al. [4] used LoRa in their air quality monitoring system to monitor particulate matter (PM). Thu et al. [1] developed a machine learning-based air quality monitoring system using LoRa over LTE-M because of a power-range trade-off. LoRa is also being used in many other applications like agro-intelligence [2], industrial automation [5], healthcare [6] etc.

B. Single Board Computers

Single-board computers (SBCs) currently on the market are strong enough to handle typical operations and workloads. In addition, single-board computers are incredibly compact. This enables them to be incorporated in devices with limited area. The computers are also highly energy-efficient, giving them an advantage in terms of power conservation. In addition to these benefits, single board computers are self-contained, making them extremely dependable in various environments. Many of these boards can be linked together to build small, low-cost clusters that resemble giant data center clusters in appearance. These clusters enable Edge/Fog Compute, in which computation is pushed out towards data sources, decreasing bandwidth requirements, and decentralizing the design. Table I lists some of the most popular low cost microcontrollers.

Advancement in digital sensor technology has made it possible to measure multiple parameters using a single sensor. However, to match the demands of an application situation, many sensors are required. Temperature, pressure, humidity, volatile organic compounds, salinity, and other characteristics, many of which are ubiquitous, are some key parameters in sensing. This brings the idea of a single node that can be used in multiple sensing applications out of the box. This has inspired us to develop iBUG, a single device with multiple sensors to measure key parameters, a LoRa node for reliable long-range data transmission, and a machine learning capability to perform in-situ processing.

TABLE II: TinyML frameworks [10]

Framework	Compatible Hardware	Output Language
TensorFlow Lite	ARM Cortex-M	C++ 11
Weka-porter	Not Fixed	C, Java, JavaScript
ELL	ARM Cortex-A, Arduino, Micro:bit	C, C++
TinyMLgen	ESP32, ARM Cortex-M	C
uTensor	mBed	C++ 11
ARM-NN	ARM Cortex-A, ARM Ethos, ARM Mali	C
STM 32Cube AI	ARM Cortex-M	C99
MicroMLGen	ESP32, ESP8288	C
CMSIS-NN	ARM Cortex-M	C99
emlearn	ESP8266, AVR ATMega	C
m2cgen	Not Fixed	C, C#, R, Python Dart, PHP, Java, JavaScript, Go
AlfES	ARM Cortex-M4, Arduino, Raspberry Pi, STM32, Windows(DLL), ATMega32U4	C
NanoEdgeAI Studio	ARM Cortex-M	C

C. TinyML

Another aspect of IoT applications is data processing. The traditional way is to carry all data to the cloud and then do the processing with a high computing device. However, data transmission consumes many resources, and the speed and reliability of an IoT application depend on it. As a result, AI-enabled edge devices are becoming popular. These devices can execute a limited set of machine learning instructions. Even with minimal capability, this local processing can help a great deal to reduce data transmission costs by sending only the data that requires further processing [7], [8]. Adding intelligence to edge devices makes them self-contained and allows them to make intelligent decisions based on the data they collect. Researchers have used the ML in edge devices in several applications like smart cities [1], traffic flow monitoring, environmental monitoring [9], etc. Machine learning inference on edge devices has a significant amount of potential, but it's still in its infancy. Various software platforms and libraries have evolved to support machine learning on edge devices. For example platforms and tools like TensorFlow Lite of Google, ELL of Microsoft, Edge Impulse converts traditional machine learning models into C/C++ code that can be executed on microcontroller powered devices. A detailed list of currently available frameworks are given in Table II.

TinyML is a machine learning paradigm for embedded edge devices with low processing power and memory (fig:1). Machine learning systems should consume only a few milliwatts of power. By merging smart power management modules, TinyML frequently enables IoT-based embedded edge devices to move to low-power systems. Hardware acceleration should be used in this system. Furthermore, in order to save electricity, the TinyML scenario's machine learning program should be as light as possible. TinyML systems should concentrate on improving accuracy by optimizing numerous machine learning models while dealing with minimal resources.

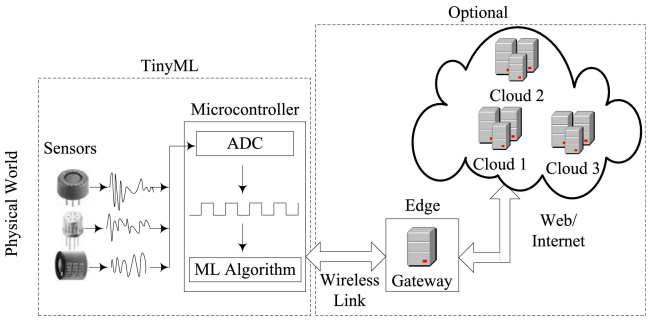


Fig. 1: Edge Machine Learning Workflow [11]

D. Short and Long Range Communication

With the development of novel environmental sensing methods, adaptive input applications (e.g., greenhouse gases, nitrogen in the soil and water), and mapping techniques, there is a higher demand for increased data rates as well as short and long-range communications [12]. iBUG enables ubiquitous protocols with short-range networks by offering Bluetooth Low Energy (BLE) and WiFi connectivity and long-range range network with LoRaWAN. WiFi is justified by its popularity and easy to deploy; however, requires higher energy consumption. Therefore, the BLE module is added for short range communication between devices with lowest power consumption. On the other hand, LoRaWAN has a small data rate, but a large coverage and low power consumption. Therefore, this protocol is appropriate for use when the solution needs to transmit a small amounts of data at very long distances ($\sim$ 3.7 miles).

III. iBUG HARDWARE DESIGN

The iBUG is a single-board microcomputer designed for those who require a complete IoT edge computing solution. To select the microprocessor for this board we considered the factors – edge AI capabilities, LoRa support, price etc. We're focusing on edge AI because we want iBUG to make real-time decisions based on the data collected by its sensors. We choose the LoRa module for long range communication. LoRaWAN is a connectivity protocol well suited to sending small payloads (such as sensor data) over great distances. Compared to rival wireless data transmission systems, LoRaWAN enables a much higher communication range with reduced bandwidths and power consumption.

The RAK11300 microprocessor was utilized in the development of iBUG. The RAK11300 WisDuo LPWAN module is a LoRa-compatible module with a 133MHz processor and a dual-core Raspberry RP2040 MCU. It can also perform cutting-edge AI. It is a low-cost CPU that supports edge AI as well as LoRaWan. Along with wifi, we've added a BLE 5 module for short-range communication. A GPS module is already installed on the board (Figure-2(a)). GPS module can be used to locate the iBUG for remote environmental sensing. A micro USB cable can be used to power the board, or a lithium-ion battery can be utilized as a power source. There is also the possibility of connecting a solar panel to the battery to charge it. A SD card slot, four I2C and GPIO connectors, and a backup battery are all found on the back of the iBUG (Figure-2(b)). The SD card slot can be utilized for both local storage and backup in the event that the data transfer fails. iBUG uses I2C ports to connect to various peripherals as needed.

The iBUG is a simple to program board. It may be configured to access and gather data from numerous sensors connected to it using the Arduino IDE. Using various platforms such as TensorFlow Lite or Edge Impulse, collected data can be utilized to train machine learning models, which can subsequently be deployed on the board to make inferences or predictions.

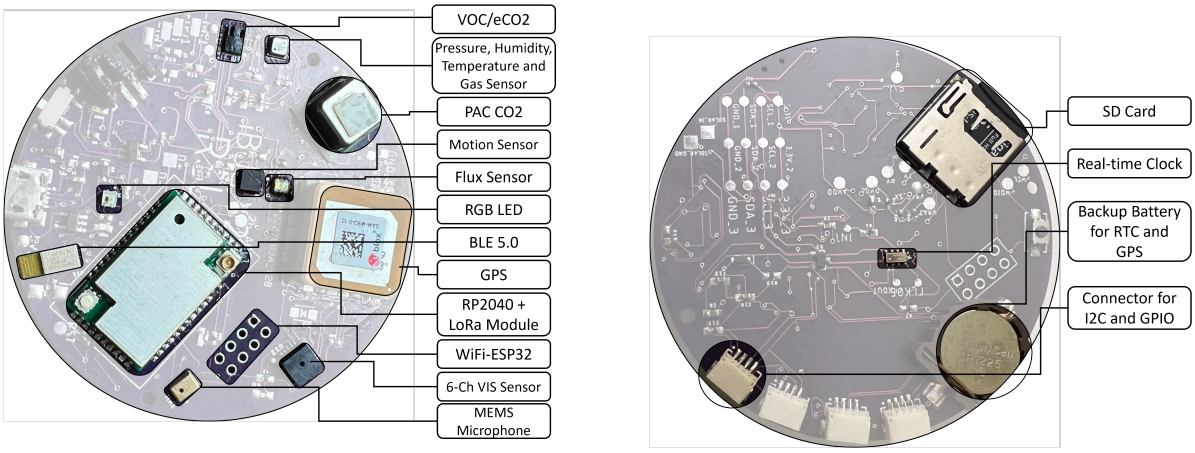
IV. PROPOSED METHOD

The motivation for this work comes from an ongoing research aimed at resource constrained machine learning problems [3]. In an effort to establish a large dataset for this research, the data collection is performed continuously for several hours a day, over a long period. In the following sections, a case study has been introduced to highlight an application of the functionalities provided by iBUG. Indoor Air Quality (IAQ) describes the interior and exterior air quality around a building structure. IAQ is affected by gases like Carbon Monoxide (CO), Carbon Dioxide (CO_2), Volatile Organic Compound (VOC) etc. VOC includes any form of carbon compound excluding CO and CO_2. Since both VOC and CO_2 are used to derive IAQ measures, the aim of this study is to predict CO_2 concentration with VOC, in realtime, by deploying a trained Machine Learning model on the iBUG processor. The BME680 sensor collects Gas Resistance values which is used as a proxy for VOC.

A. Data Acquisition and Processing

The dataset used was collected using iBUG's SME4x CO_2 sensor and BME680 Gas Resistance (R) sensor. The CO_2 concentration ($CONC$) is measured in Parts Per Million (PPM), where as the Gas Resistance (R) is measured in Kilo-Ohms ($K\Omega$). The data collection experiment was conducted in a university laboratory, consisting of a single occupant. The data collection was carried out over a time period of 11 hours and 20 minutes. The sensors read each quantity at a five-second interval. Figure-3 and 4 shows the $CONC$ and R readings over time, from the two settings. As shown in the two figures, $CONC$ and R have a negative correlation which is calculated from the collected dataset as -0.76. It is recommended in the sensor guidelines to run the sensor for 30 minutes in the desired mode before considering any readings. As shown in the two figures, there is a sharp drop in the $CONC$ level during the initial data collection step. However, per the sensor guidelines, the data points of the first 30 minutes were still discarded.

The dataset $D = \{\vec{x}, \vec{y}\}$, is a set of feature and target vectors, where $\vec{x} = \{R\}$ and $\vec{y} = \{CONC\}$. The first 80% of the data was used as the training set, and the remaining 20%



(a) Front (b) Back

Fig. 2: Built-in components of iBUG: Volatile organic compounds (VOC); estimated CO_2 (eCO_2); photoacoustic cell (PAC); red, green, and blue (RGB); Bluetooth Low Energy (BLE); Global Positioning System (GPS); Channel (Ch); Visible (VIS); Micro-electromechanical systems (MEMS); Secure Digital (SD)

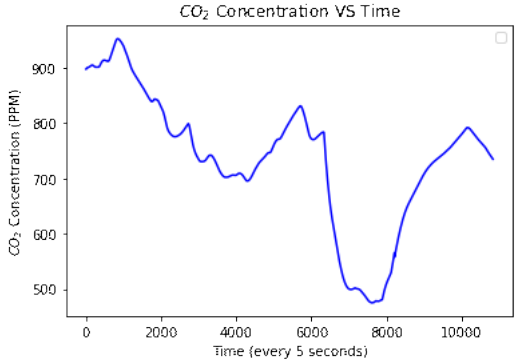


Fig. 3: CO_2 Concentration(Parts Per Million) recorded in a university laboratory room

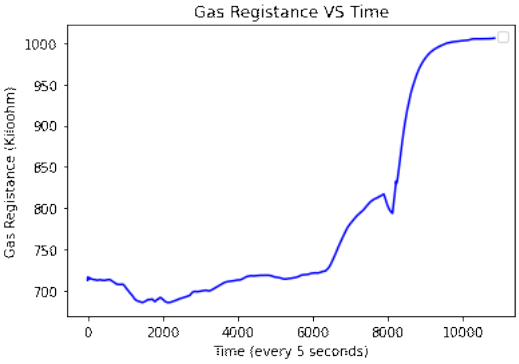


Fig. 4: Gas Resistance (Kilo-Ohms) recorded in a university laboratory room

was used for model testing. The last 20% of the training set was further separated for validation during training.

B. Model Overview

In this paper, two distinct Deep Learning (DL) architectures were implemented, namely- Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM) network. The implemented ANN consists of an input layer, followed by three hidden layers and an output layer. ANN is feed-forward in nature, that is the connection between the neurons in the consecutive layers does not form a cycle. This means information flows through the network in a single direction. The second implemented model, LSTMs are a descendant of Recurrent Neural Networks (RNN) and more details about its architecture can be found in [13]. In contrast to a feed-forward network, a standard RNN forms a directed graph amongst it's neurons, based on temporal characteristics. Due to this RNN and its variants, we are able to take into account the temporal properties of a time series data. RNN consists of a single Neural Network(NN) layer, whereas LSTM consists of four NN layers, interacting with each other in a specific way. In Figure-3, the lines carry input/output vectors, to/from a node. The vector C_{t-1}-C_t represents the cell state. The LSTM network has the ability to edit information to the cell state, which is imposed by structures called gates. Gates allow information to pass depending on the values of particular operations. A gate consists of a sigmoid neural network (σ-NN) layer and a pointwise multiplication operation. The output of the σ layer is a binary value, where 0 indicates that information will not pass and 1 indicates that all information shall pass. LSTM consists of 3 gates, which it uses to interact with the cell state, namely- forget gate; input gate; and output gate.

Depending on the output from time step $t-1$, h_{t-1} and input from time step t, x_t, the output gate decides decides

which information to discard from cell state C_{t-1}. The output gate value is calculated as shown in Equation-1, and $f\epsilon$ (0,1). The terms W_f and b_f represents the weight and the bias of the forget gate respectively.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{1}$$

With the aid of the input gate and a tanh layer, information is added to the cell state. The input gate generates the output i_t and the tanh layer gives an output in the form of a candidate state $\hat{C}_t$. The mathematical calculation is exhibited in Equation-2 and 3, where $i\epsilon$ (0,1); (W_i, b_i) and (W_c, b_c) are the weights and biases of the input gate and the tanh layer respectively.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\hat{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{3}$$

The new cell state C_t is derived by using pointwise addition and multiplication operations on the gate outputs with the old cell state, C_{t-1} and the candidate state $\hat{C}_t$. The mathematical operation is exhibited in Equation-4.

$$C_t = f_t C_{t-1} + i_t \hat{C}_t \tag{4}$$

Next, the output h_t is calculated. The cell state C_t is passed through a tanh layer and a pointwise multiplication operation is carried out between the result of the tanh layer and the output of the output gate, o_t. The calculations are shown in Equation-5 and 6, where W_o and b_o are the weights and biases of the output gate. The final output from he LSTM gets passed to a Dense layer, which ensures that all the predicted outputs $\hat{y}$ are mapped to the output layer of the implemented model properly.

Equation-7 denotes Mean Squared Error (MSE), which is used as the loss function for both the models. The loss function is minimized using the training and validation dataset, to find the optimal set of weights, which in turn indicates the optimal connection between neurons. In this paper, the RMSprop optimizer was used to minimize the loss function. RMSprop is a form of first order stochastic gradient descent optimization technique. The first order nature indicates that it uses a first derivative. RMSprop uses an adaptive learning rate, which means the learning rate changes over time. It achieves this by normalizing the gradient using a moving average of squared gradients. This normalization step in turn decreases the step size for large gradients and increases it for small gradient.

The two models were separately trained offline, using the same training and validation set. The test set was used to make offline prediction of CO_2 concentration ($CO\hat{N}C_{off}$) and to calculate the offline model accuracy for each model architecture. Using tensorflow-lite and the Arduino IDE each of the trained models were separately deployed inside iBUG. For each model variant, the realtime or online prediction of CO_2 concentration ($CO\hat{N}C_{on}$), along with the corresponding actual $CONC$ from the CO_2 were obtained in order to calculate online prediction accuracy.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t tanh(C_t) \tag{6}$$

$$\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2. \tag{7}$$

C. Deployment On iBUG

We converted the model using TensorFlow Lite to be deployed on the RAK11300 microprocessor. First, the model was converted to a lite version. Then quantization method was applied to the model to reduce the model size. Quantization reduces the model size, keeping the performance almost unchanged. Next, the quantized model was converted to c header file using a Linux command which contains an array of the prediction model. We used EloquentTinyML, a library, to reduce some of the boilerplate code of TensorFlow Lite. It gives a clean and straightforward way to code and deploys in iBUG using Arduino IDE. Anyone with a bit of knowledge of c programming should be comfortable with it. Finally, the model was flashed into the iBUG and tested in a condition similar to the data collection environment.

V. Results and Discussion

To test the realtime prediction capability of the deployed models, the iBUG board was placed in an environment similar to the one from where the training data was collected. The BME and SME4x sensors collects the R and $CONC$ values at five seconds interval, and the deployed model predicts the corresponding CO_2 concentration in realtime. Figure-5 and Figure-6 shows the actual versus predicted value of $CONC$ for the deployed ANN and LSTM model respectively. The actual and predicted $CONC$ data are time series in nature and it lacks any form of formal trend. Therefore, simple exponential smoothing operation were applied on the collected and predicted $CONC$ before generating the plots. From the figures, it can be observed that the results generated by the ANN model is better at capturing the trend of the actual $CONC$, compared to the one predicted by the LSTM model. The is assessment of the performance is also supported by the realtime prediction accuracy of both the models. The ANN has a realtime prediction accuracy of 74.01%, where as the LSTM has a realtime prediction accuracy of 69.21%. The offline prediction accuracy of each model were calculated, where the ANN model had an accuracy of 85.02%, whereas LSTM had an accuracy of 84.98%. The accuracy of both the models are comparable when used for offline prediction. Although, the offline results of the model out performs it's realtime counterpart, it is to be noted that objective of this experiment was to exhibit the realtime prediction capability of the iBUG board, with a relatively small dataset. However, a larger dataset with more variability would allow the model to be better trained. For this paper, the testing area had a stable

environment with the number of people present varying from two to six. In the future, further parameter and model tuning will be performed in order to reap better accuracy when it comes to realtime prediction.

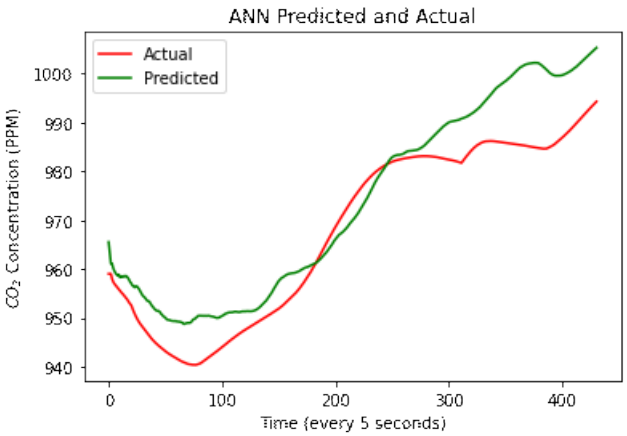


Fig. 5: Actual and Realtime Predicted CO_2 Concentration by ANN

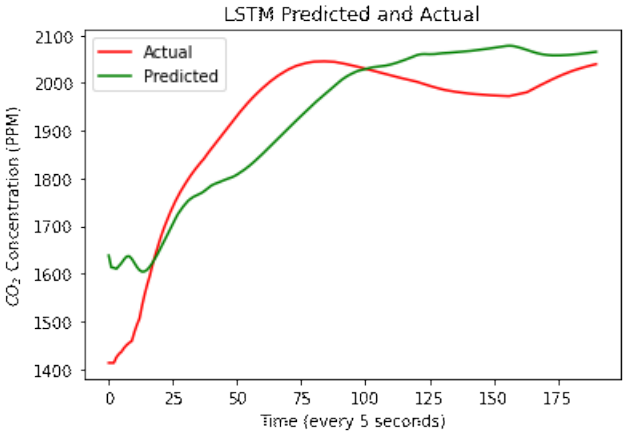


Fig. 6: Actual and Realtime Predicted CO_2 Concentration by LSTM model

An application of the realtime prediction could in it's current form could be to detect anomalies. The predictions can be treated as estimates, where a threshold value can be used to detect any abrupt changes in the CO_2 concentration. Another application of this board, is that it can be treated as a go-to device for both researchers and students so that they can focus on the primary problem they're seeking to address. For example, researchers working in environmental sensing or remote sensing can use this board to quickly get started on their project without worrying about integrating all of the components and sensors needed for data collecting and real-time decision making. Students will likewise benefit in the same way. This gives them everything they need in one package to learn and play with real-time machine learning. Students are already utilizing it in a graduate course at the University of New Hampshire called *Real-time Machine Learning* (ECE 992), which focuses on machine learning on embedded devices. Our code will be available as libraries so that anyone can easily train their model with data where they want to deploy iBUG. We used ANN and LSTM as example, but any other deep learning method can be used if necessary.

VI. Conclusion

The growth of the Internet of Things (IoT) and intelligent devices has resulted in a tremendous increase in data production. To lower data transfer costs, computation is being shifted to the periphery. Real-time machine learning on edge devices has emerged as a powerful tool in this scenario. In this paper, we demonstrate a real-time machine learning capable board that may be used for a number of applications, including environmental sensing. We also included a case study that shows how machine learning models can be utilized as a feasible sensor replacement. It will be thrilling to discover many more use cases and work on increasing performance in the future.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant SitS-192096.

References

[1] M. Y. Thu, W. Htun, Y. L. Aung, P. E. E. Shwe, and N. M. Tun, "Smart air quality monitoring system with LoRaWAN," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, 2018, pp. 10–15.

[2] A. Valente, S. Silva, D. Duarte, F. Cabral Pinto, and S. Soares, "Low-cost LoRaWAN node for agro-intelligence IoT," *Electronics*, vol. 9, no. 6, p. 987, 2020, number: 6 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/9/6/987

[3] B. Coffen and M. Mahmud, "Tinydl: Edge computing and deep learning based real-time hand gesture recognition using wearable sensor," in *2020 IEEE International Conference on E-health Networking, Application Services (HEALTHCOM)*, 2021, pp. 1–6.

[4] J. Park, Y. Oh, H.-h. Byun, and C.-k. Kim, "Low cost fine-grained air quality monitoring system using LoRa WAN," in *2019 International Conference on Information Networking (ICOIN)*, 2019, pp. 439–441, ISSN: 1976-7684.

[5] E. Sisinni, D. F. Carvalho, P. Ferrari, A. Flammini, D. R. C. Silva, and I. M. D. Da Silva, "Enhanced flexible LoRaWAN node for industrial IoT," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[6] M. Haghi Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, "A systematic review of IoT in healthcare: Applications, techniques, and trends," *Journal of Network and Computer Applications*, vol. 192, p. 103164, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521001764

[7] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, Jan. 2018, conference Name: IEEE Network.

[8] M. T. Yazici, S. Basurra, and M. M. Gaber, "Edge machine learning: Enabling smart internet of things applications," *Big Data and Cognitive Computing*, vol. 2, no. 3, p. 26, 2018, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2504-2289/2/3/26

[9] B. Arshad, R. Ogie, J. Barthelemy, B. Pradhan, N. Verstaevel, and P. Perez, "Computer vision and IoT-based sensors in flood monitoring and mapping: A systematic review," *Sensors*, vol. 19, no. 22, p. 5012, 2019, number: 22 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/5012

[10] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, no. 3, pp. 4–18, 2020, conference Name: IEEE Circuits and Systems Magazine.

[11] D. L. Dutta and S. Bharali, "TinyML meets IoT: A comprehensive survey," *Internet of Things*, vol. 16, p. 100461, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660521001025

[12] N. Islam, M. M. Rashid, F. Pasandideh, B. Ray, S. Moore, and R. Kadel, "A review of applications and communication technologies for internet of things (iot) and unmanned aerial vehicle (uav) based sustainable smart farming," *Sustainability*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/4/1821

[13] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-Based Model to Forecast Stock Prices," *Complexity*, vol. 2020, p. e6622927, Nov. 2020, publisher: Hindawi. [Online]. Available: https://www.hindawi.com/journals/complexity/2020/6622927/