Machine Learning Models for Prediction of Metal Ion Concentrations in Drinking Water

Nehpal S. Shekhawat[1], Sangmin Oh[1], Cristinel Ababei[1], Chung Hoon Lee[1], and Dong Hye Ye[2]
[1]Electrical and Computer Engineering, Marquette University
[2]Computer Science, Georgia State University
Email: {nehpal.shekhawat, sangmin.oh, cristinel.ababei, chunghoon.lee}@marquette.edu, dongye@gsu.edu

***Abstract*—We present an investigation of several machine learning (ML) models developed to predict the copper (Cu) and lead (Pb) ion concentrations in drinking water. The system where this prediction is employed is based on a microwave block loop gap resonator (BLGR), which surrounds a glass tube with drinking water passing through it. The resonator is coupled to a vector network analyzer (VNA), which collects reflection coefficient (S11) measurements over a 100 MHz - 6 GHz frequency range. It is these S11 measurements, in raw format or compressed using various signal processing techniques, that are used as input into the ML models. Our investigation looks at new convolutional neural networks (CNN) and deep neural networks (DNN) models because such models can easily be deployed on IoT microcontroller devices using tinyML technologies. Extensive simulations using real data demonstrate that DNN models that use as input features essential spectral information created from S11 traces provide performance comparable to that of CNN models but at much shorter training times and significantly smaller model sizes.**

***Index Terms*—ppb-level metal ions; prediction; machine learning; CNN; DNN; spectrogram;**

I. Introduction

Increased scarcity of water resources and increased pollution threaten water availability and quality worldwide. Global climate change and human activities (farming and industrial) are prime factors affecting significantly water resources. These increasingly adverse factors only amplify the need for continuous monitoring of water resources to make sure that water quality meets minimum quality standards. Particularly, monitoring drinking water to detect the presence of pollutants, chemicals, or heavy metal ions is crucial in ensuring safety of the population.

In this paper, we investigate deep machine learning (ML) models, for the prediction of both Cu and Pb ion concentrations (measured as ppb) in drinking water. The primary objective of this investigation is to identify the model that provides the best performance (as estimation error) and yet is as small as possible - in terms of: 1) model size on disk (which impacts the ability to use it on embedded devices that are memory constrained as well as the inference runtime), 2) format and size of the input provided to the model, and 3) training runtime. More specifically, we evaluate convolutional neural networks (CNN) and deep neural networks (DNN) models, which we train and test with real data. Our experiments showed that DNN models that use essential spectral information created from S11 traces as input features provide performance comparable to that of CNN models but at much shorter training times and significantly smaller model sizes.

II. Previous Work

There has been significant work on various methods and technologies to monitor drinking water. Generally, one can classify these methods into two categories: those that require direct contact with the water [1]–[3] and those that are contactless [4]. We are not interested in this work in the approaches that require direct contact with the water and therefore we do not discuss related literature on that type of approach here. Our own approach is in the second category, that of contactless approaches.

In our recent previous work [4], we introduced a new sensor system (will be described later on, in the next section) to estimate the Pb ion concentration as ppb - which does not need any contact with the measured water. The sensing component of that system uses a novel microwave block loop gap resonator (BLGR) coupled with a vector network analyzer (VNA), which collects reflection coefficient, S11, measurements as raw data that then is used as input into an SVR machine learning model, which was trained to do regression for the Pb ion concentration. However, the SVR model's performance assessed as estimation error was 13%. In this paper, we continue that work by looking at new additional ML models and frame the problem statement as a classification problem rather than a regression problem.

The idea of using a resonator is that the presence of ions in the water can shift the resonant frequencies of the resonator, which is captured by the reflection coefficient S11 measurements. The resonant frequencies shift depends on several factors, including the concentrations of ions and the background of the water sample. The study in [5] used an microwave sensor array to investigate resonant frequencies for water mixed with several contaminants including nitrate, phosphate, ammonium, and Pb. The study is a characterization of resonant frequencies and not a concentration measurement system as in our case; in addition, their experimental setup does actually involve contact between water and the sensor array. The sensor array was expanded in [6] to employ machine learning to predict the presence of phosphate, Pb, mercury, and chromium. The study investigated several machine learning classifiers (i.e., SVM, K-nearest neighbors, KNN, and random forest, RF) and reported that RF provided the best overall

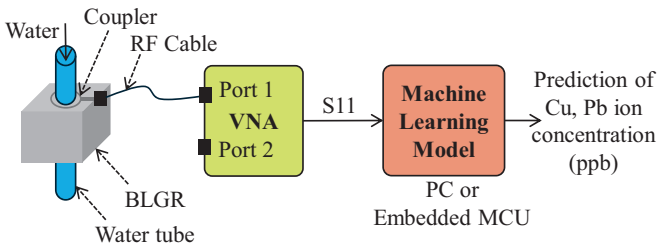

Figure 1. System level diagram of the sensor system from [4] where the ML models investigated in this paper are used. Note that there is no direct contact with the tested water.

accuracy of 81.3% for classification. Their sensor system is designed to use the classifier to discriminate between four classes corresponding to the four different contaminants. In contrast, in our work, we develop ML models that predict the concentration level of a given contaminant. Other previous studies that used microwave resonators include [7]–[9], but, they focused on sensing glucose, phosphate and nitrate, or ammonia and iron concentrations in water and did not employ machine learning techniques.

III. Problem Statement and Datasets

A. Problem Statement

The problem that we are trying to solve in this work is the lack of small-enough ML models for prediction of Cu and Pb ion concentrations in drinking water. We want such models to be small-enough in the sense that their size should be small to allow them to be deployed on microcontrollers, which have very limited memory capacities, yet with high enough prediction performance to warrant their deployment on such devices. We look at this problem in a specific context - that of using such ML models in our own experimental sensor system that we have introduced in our recent work [4].

To present a clear picture of our problem statement, we must provide details about the sensor system where the investigated models will be deployed in practice. A simplified block diagram of the sensor system is shown in Fig. 1. The block loop gap resonator (BLGR) has the glass tube go through it as shown in the figure. The BLGR has attached to it the inductive coupling loop, which is fabricated on a printed circuit board, and connected to Port 1 of the VNA. The VNA records the reflection coefficient, S11, over a 100 MHz - 6 GHz frequency range. S11 measures the ratio of the reflection from Port 1 to the drive signal at that port (see Fig. 1). Therefore raw S11 measurements are in the frequency domain, and they include values of both the magnitude and phase over the stated frequency range. In our sensor system however, only S11 amplitude values are used as input into the ML models. An example of such an S11 measurement is shown in Fig. 2 - where the collected 20,000 S11 magnitude values are downsampled to only show 2,500 values spanning the frequency range of 100 MHz - 6 GHz.

As indicated in Fig. 1, the S11 measured data is passed as raw input to the ML models, which mus predict metal ion

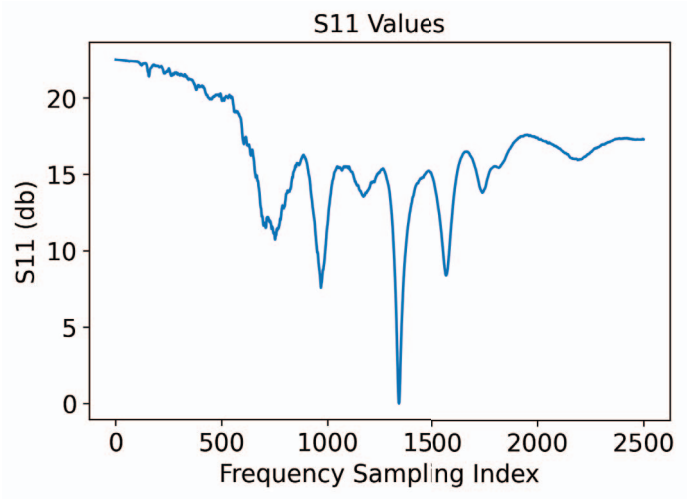

Figure 2. Example of input used for the CNN model: this S11 plot represents a flattened sequence of 2,500 values of S11 sampled from the frequency range 100 MHz - 6 GHz.

concentrations. While the previous SVR model was executed on a PC, we are interested in deploying it on embedded devices, which are resource constrained - both in terms of memory and computational power - and therefore we are especially interested in arriving at new ML models that are small yet high performance for our application so that they can easily be deployed on resource-constrained microcontroller units (MCUs). We also note that, in our previous work we used SVR models to do *regression*, here we focus on developing simpler models to do *classification*. We focus on a classification problem framing because of the feedback from industrial partners who are interested more in predicting several distinct ppb ion levels, which represent the classes in the classification problem formulation. Therefore, we cannot compare directly our new models to SVR, but, we will compare against SVM models. For a more detailed description of the entire sensor system from Fig. 1 and the theory behind it, see our recent previous work [4], which looked at Pb ions only and used SVR models for prediction.

B. Datasets Description

All ML models investigated in this paper are supervised models, whose training require datasests. Because currently there are no publicly available datasets for our specific problem, we have collected our own datasets. In a laboratory set-up, we used the sensor system from Fig. 1 to collect raw S11 measurements (like those shown in Fig. 2) using water samples with known levels of ion concentrations, specifically 0,1,5, and 10 ppb levels. We collected 12,000 measurements for two datasets that we created - one for Cu and one for Pb. These two datasets are collected separately for each of the two metal ions. The datasets will be used for training and testing of all models investigated. The train/test split ratio that we use is 0.7/0.3 for each of the two datasets because it is a commonly used split ratio in the literature.

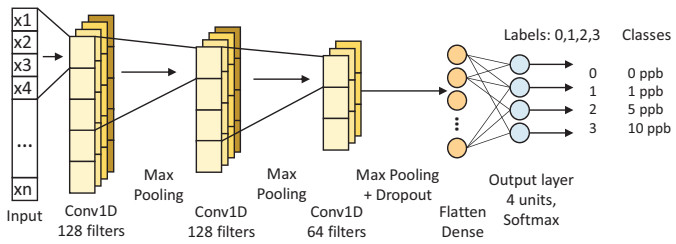

Figure 3. CNN model studied in this paper. $n = 2,500$ is the number of S11 magnitude values that are fed as one input into the model; an instance of such values is shown in Fig. 2.

IV. Investigated Models

A. SVM Method with S11 Values as Input

First, we investigate support vector machines (SVMs) as a popular supervised model because of its popularity in previous literature. While in our previous work [4] we used support vector regression (SVR) for prediction of Pb ion concentration, in this paper, we frame the problem as a classification problem. Therefore, for classification, we investigate first a support vector machine (SVM) model. SVM has the advantage of shorter training times, but, its disadvantages include larger model-size and slow prediction. The number of parameters of an SVM increases linearly with the increase of input size. In contrast, neural networks offer much higher prediction speed and smaller model-size. Generally, neural networks have been shown to offer better performance/accuracy, especially for large datasets; hence, it is suggested to use neural networks for high-dimensional large data sets and SVMs for low-dimensional small data sets.

To investigate SVMs, we have created and trained a multi-class nonlinear SVM classifier using the Python library *sklearn.svm* [10], which by default uses *kernel='rbf'* (radial basis function, RBF). The SVM model achieves in our Python simulations 100% accuracy. However, currently SVM models and algorithms are not directly supported by tinyML technologies such as TensorFlow Lite for Microcontrollers [11], which we intend to use for deploying ML models on microcontroller devices. Such tinyML technologies do support CNN and DNN models and that will enable us to deploy them to IoT microcontroler devices. Therefore, it is CNN and DNN models that are really our main focus in this paper.

B. CNN Model with S11 Values as Input

IN this section, we propose to investigate is the convolutional neural network (CNN) model. CNN models have been used traditionally in computer vision applications, but, recently they have been applied successfully in virtually all application domains. One of the most challenging aspects of developing ML models such as CNN models include: 1) how to define the network architecture (i.e., number of layers, number of units per layer, number of filters, batch size, number of epochs used for training, etc.) and 2) how to frame the problem at hand to make it amenable to solving it with the developed model.

Table I
Characteristics of the investigated models.

Characteristic	CNN	DNN1	DNN2
Total params	153796	52164	10788
Model size	1850 KB	643 KB	158 KB

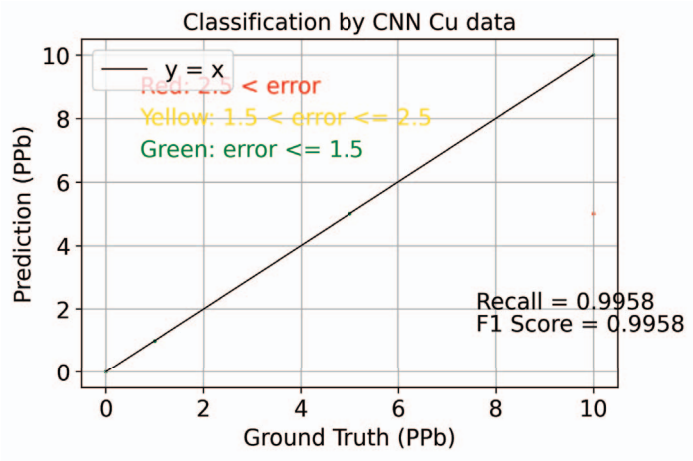

Figure 4. CNN model: true and predicted Cu ppb values for all datapoints from the testing portion of the Cu dataset.

In our case, we developed the studied CNN model based on: our previous experience with such models, inspiration from recent literature dealing with CNN models, and on an empirical trial-and-error process during which several different values for model parameters have been investigated. The final CNN model used in this paper is shown in Fig. 3. In addition to the model architecture parameters illustrated in this figure, we used during training: $sparse\_categorical\_crossentropy$ as loss function, $Adam$ as an optimizer, and a batch size of 32. The model summary is shown in Table I, which shows the total number of parameters and the model size (in KB) for all models presented in this paper, for comparison purposes.

We frame the problem as follows: given the input as S11 measurements (20,000 values spanning the frequency range of 100 MHz - 6 GHz), use the CNN model to *perform classification*. Therefore the output layer has four units (i.e., neurons) corresponding to four different labels {0, 1, 2, 3} which encode the four different ppb levels we work with in this paper {0 ppb, 1 ppb, 5 ppb, 10 ppb}. These output units use *Softmax* activation function. To keep the input size to more reasonable, smaller values, but without impacting negatively the quality of the prediction, we undersample (i.e., downsample) the S11 values of a given ppb concentration measurement by 8. That is, we only use 2,500 samples (i.e., $n = 2,500$ in Fig. 3) as input into the CNN model.

We train two versions of the CNN model: one for Cu and one for Pb. The training process was done for a number of 25 epochs using the datasets described in the previous section. Once models are trained, they are used to make inferences on the testing portions of the datasets. The result of this testing

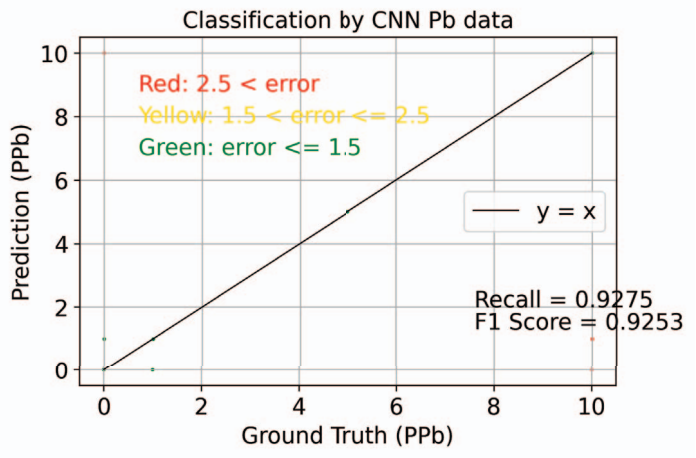

Figure 5. CNN model: true and predicted Pb ppb values for all datapoints from the testing portion of the Pb dataset.

Table II
PERFORMANCE COMPARISON OF THE INVESTIGATED MODELS.

Performance	CNN	DNN1	DNN2
F1 Score Cu	0.9958	0.9277	0.9983
Accuracy Cu	0.9958	0.9292	0.9983
Precision Cu	0.9959	0.9447	0.9983
Recall Cu	0.9958	0.9292	0.9983
F1 Score Pb	0.9253	0.926	0.9126
Accuracy Pb	0.9275	0.9286	0.9158
Precision Pb	0.937	0.9381	0.9246
Recall Pb	0.9275	0.9285	0.9158

is shown in Fig. 4 and Fig. 5. The overall performance of the trained models is assessed with the help of several common metrics, which are obtained using *classification_report()* from scikit-learn library. These metrics are listed in Table II.

C. DNN Model with Spectrogram of S11 Series as Input

Another model investigated in this paper is motivated by the desire to decrease the size of both the CNN model and the input fed into the CNN model presented in the previous section. That is because even when we used undersampling by 8 and reduced the number of S11 values to 2,500 fed as input at any one time into the model, the model was too large, especially for deployment on edge devices.

As such, in this model development, we started by reducing the 2,500 number for the input feature size. We propose to do that by constructing *spectrograms* from S11 raw data measurements. The idea is to treat or interpret the S11 series (like that shown in Fig. 2) as if it were an audio signal and generate for it a spectrogram of a size that is much smaller than 2,500. This can be done easily with existing digital signal processing libraries available for Python. In this paper, we use *tfio.audio.spectrogram* to generate spectrograms that have 20x17 = 340 values. One can easily tune several parameters provided by *tfio.audio.spectrogram* to change the

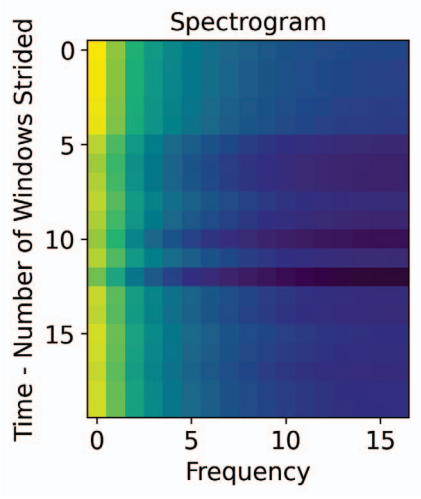

Figure 6. Spectrogram generated for the S11 series from Fig. 2.

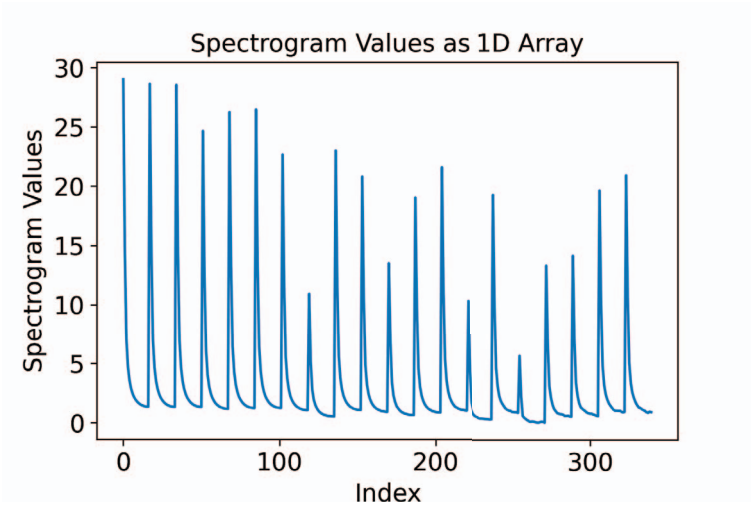

Figure 7. The spectrogram from Fig. 6 flattened by reading the values of the spectrogram entries row by row and column by column.

size of the spectrogram. In our case, the 20x17 was selected based on the total number of 340 values that was small enough, yet provided good model performance (discussed later). This spectrogram size was obtained with the following specific parameters *tfio.audio.spectrogram*: assumed sampling frequency of 16 kHz, size of FTT (fast Fourier transform) of 32, size of window of 256, and size of hops between windows of 128.

As an example, the spectrogram generated for the S11 series (with 2,500 values) from Fig. 2 is shown in Fig. 6. Normally, a spectrogram displays changes in the frequencies in an audio signal over time. Spectrograms are generated by sweeping the series signal with a moving window, with a stride that keeps consecutive windows at a certain desired overlap. The length of the stride determines the size of the step with which the sliding window moves over the signal series. The total number of strides gives the number of rows in Fig. 6. For each stride,

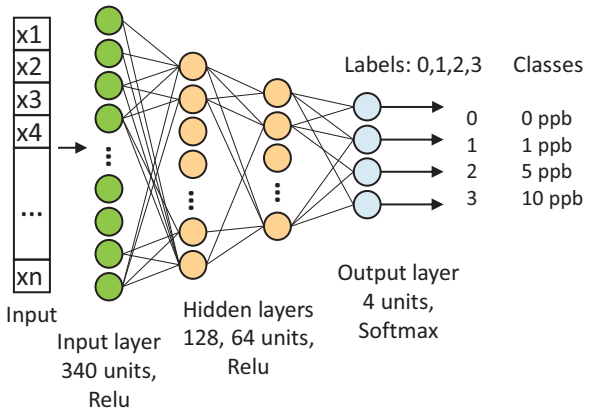

Figure 8. DNN1 model studied in this paper. $n = 340$ represents the number of values in a spectrogram that is fed as one input into the model; an instance of such values is shown in Fig. 7.

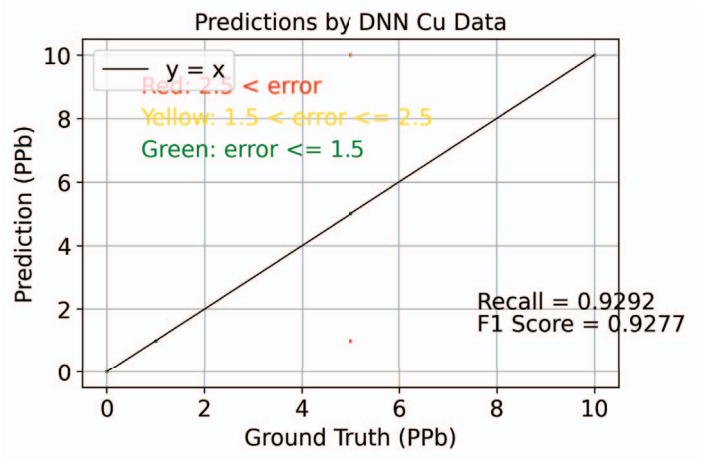

Figure 9. DNN1 model: true and predicted Cu ppb values for all datapoints from the testing portion of the Cu dataset.

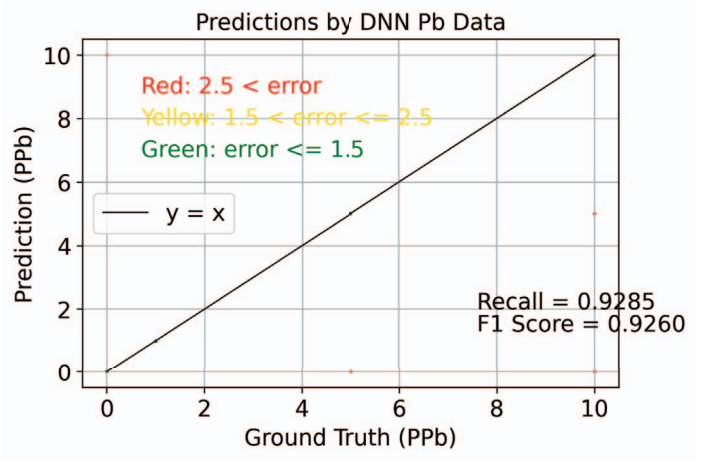

Figure 10. DNN1 model: true and predicted Pb ppb values for all datapoints from the testing portion of the Pb dataset.

the spectrogram algorithms uses FFT to find the frequency content within the moving window. This information is shown as the individual values in each row Fig. 6. The selected number of such values gives the number of columns in Fig. 6. All values inside the spectrogram are indicated on a third dimension with variable brightness or color - to show the amplitude of frequency content. This spectrogram - which now is reduced to only 340 numerical values - is precisely what will be used as input into the second ML model, which is a deep neural network (DNN) model.

To be able to easily feed the spectrogram as input to the DNN model, we first flatten the 2D spectrogram and save it as a simple 1D series of values; for example, the flattened 1D version of the spectrogram from Fig. 6 is shown in Fig. 7. This flattened 1D array of values is fed as input into the DNN model (referred to as DNN1 model in this paper) shown in Fig. 8. This model is also developed to perform classification, just like the previous model. The training of the model from Fig. 8 was done with: $sparse\_categorical\_crossentropy$ as loss function, $Adam$ as an optimizer, and a batch size of 32. The model is also summarized in Table I to compare it with the other studied models. This model also was trained/tested with a 0.7/0.3 split for both Cu and Pb datasets. The training process was done for a number of 250 epochs, which takes a much shorter training time than the 25 epochs of the previous model. The results of the testing done afterward are shown in Fig. 9 and Fig. 10. Performance measures are reported in Table II as well for comparison.

V. DNN Model with Essential Spectral Information of S11 Series as Input

To further reduce both the model and input sizes without degrading model performance, we develop here our final model. The objective was to reduce even more the input size into the model; while the input size of the DNN1 model presented in the previous section is significantly smaller than that of the CNN model, we wanted to reduce it even further. Working with smaller spectrograms turned out to degrade performance. Therefore, we replace the spectrogram with a completely different presentation of the input. Specifically, we again treat the S11 series as if it were an audio signal, but, this time we generate from it a number of what we call *essential information* values or features, that capture in a sense the "essence" of the S11 trace. For that, we employ a digital signal processing library - called *processing blocks* - that is publicly available from Edgeimpulse [12]. The *spectral analysis* portion of this library can be used to generate a desired number of features very efficiently (using a function called *generate_features()*) based primarily on the power spectral analysis of the given signal, which in our case is the S11 series.

The arguments passed to the *generate_features()* function directly determine how many features are generated. We tuned these parameters to the values that turned out to provide the best results with the smallest number of features, which in our case is 133. The two most important parameters are:

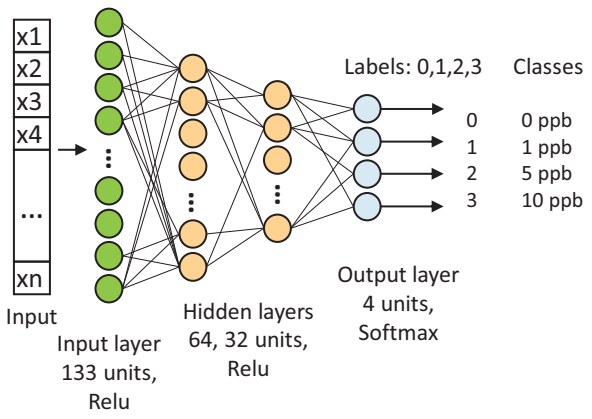

Figure 11. DNN1 model studied in this paper; $n = 133$.

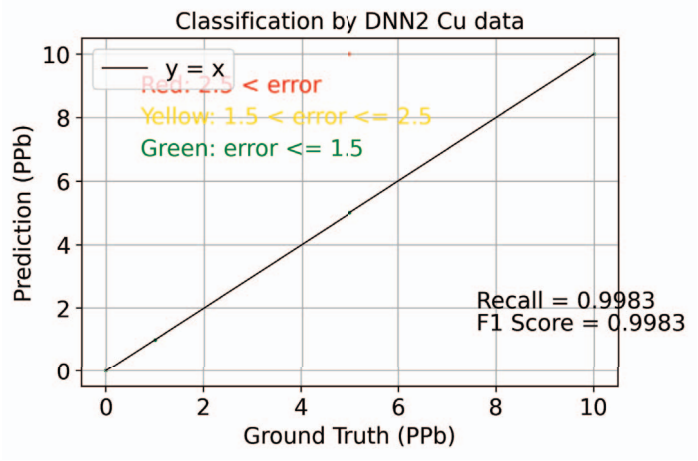

Figure 12. DNN2 model: true and predicted Cu ppb values for all datapoints from the testing portion of the Cu dataset.

sampling_freq = 62.5 (i.e., the sampling frequency of the data) and analysis_type = 'FFT' (i.e., analysis type fast Fourier transform). The 133 features generated include: *{RMS (root mean square), Skewness, Kurtosis, Spectral Skewness, Spectral Kurtosis, Spectral Power 0.12-0.37 Hz, Spectral Power 0.37-0.61 Hz, Spectral Power 0.61-0.85 Hz, ..., Spectral Power 30.4-30.64 Hz, Spectral Power 30.64-30.88 Hz, Spectral Power 30.88-31.13 Hz, Spectral Power 31.13-31.37 Hz}*.

This array of values is fed as input into the DNN model (referred to as DNN2 model in this paper) shown in Fig. 11. Similarly to previous models, this model is also developed to perform classification. Its training was done with: *sparse_categorical_crossentropy* as loss function, *Adam* as optimizer, and a batch size of 32. The model is also summarized in Table I to compare it with the other studied models. Like the other models, this model also was trained/tested with a 0.7/0.3 split for each dataset. The training process was done for a number of 300 epochs. The result of the testing is shown in Fig. 12 and Fig. 13. Performance metrics are also reported in Table II too.

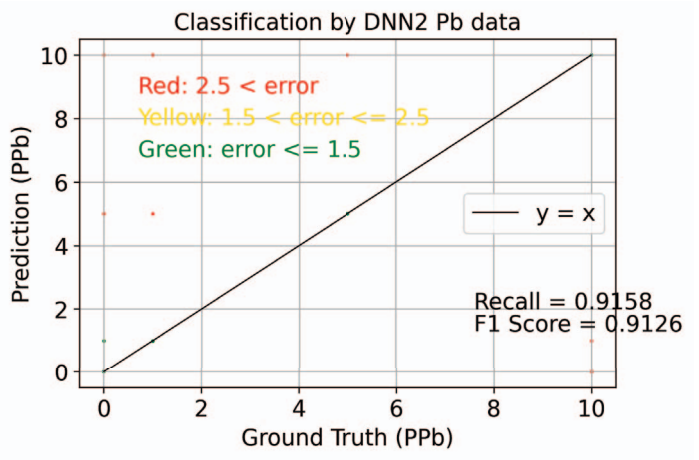

Figure 13. DNN2 model: true and predicted Pb ppb values for all datapoints from the testing portion of the Pb dataset.

VI. Discussion

Based on the characteristics of the developed models (listed in Table I) and on their performance (summarized in Table II) reported in the previous sections, we note that DNN model used with essential spectral information of S11 series as input - provides the best desirable model characteristics (i.e., small size and hence memory footprint) and comparable performance. Its size at around 157 KB is small enough to easily fit even on tightly memory-constrained microcontrollers, such as STM32L073RZ [13], which are intended particularly for edge/IoT applications.

In this paper, we focused on models for problems formulated or framed as *classification* problems - where the four classes represent the four different ppb levels. That is because the real application where these models will be deployed is a water sensor system (for residential homes), where for simplicity we would only be interested in whether the ppb level is less than a safe threshold or not - which best fits a simpler binary classification problem. Here, we investigated a four classes problem to get better insights into the capability of such models.

While we were successful at arriving at DNN models that are sufficiently small and with good performance, one can see some limitations/disadvantages of the proposed models. Specifically, we need additional processing capability to be done on the microcontroller device - that includes computations such as FFT and extraction of the essential information. This additional processing will incur increased inference delays.

VII. Conclusion and Future Work

We investigated the use of ML to predict the concentration of Cu and Pb ions (as ppb) in drinking water based on S11 measurements collected with a vector network analyzer connected to a microwave block loop gap resonator that has a glass tube with drinking water passing through it. Specifically, we developed CNN and DNN models and used as input either

the S11 values directly or various transformations of these values to reduce the number of inputs into the model and therefore model size. The problem of ppb level prediction was formulated as a classification problem. Simulation experiments indicated that DNN models that use as input features essential spectral information created from S11 traces provide performance comparable to that of CNN models but at much shorter training times and significantly smaller model sizes.

In our future work, we plan to investigate whether and to what extent only S11 measurements from certain frequency sub-bands impact primarily the model outcome as predictions. If that were the case, then, we would only need to concentrate the S11 measurements (to store into our datasets or to use at inference time) in those frequency sub-bands only rather than on the entire frequency range of 100 MHz - 6 GHz. In that way, we could reduce again the input size into our models while, reducing needed storage, speeding up real-time measurements, and focusing on only the data that matters most importantly. We plan to use layer-wise relevance propagation (LRP) techniques to identify such frequency sub-bands.

REFERENCES

[1] J. Mahato, C. R. Raj and K. Biswas, "Low-Noise Potentiostat Circuit for Electrochemical Detection of Heavy Metals or Metalloids," *IEEE Trans. on Instrumentation and Measurement*, vol. 71, pp. 1-9, 2022.

[2] D. Goldoni, L. Rovati and L. Selmi, "Toward Continuous Nano-Plastic Monitoring in Water by High Frequency Impedance Measurement With Nano-Electrode Arrays," *IEEE Sensors Journal*, vol. 23, Sep. 2023.

[3] J.-K. Wu et al., "Polarization-Based Gigahertz Near-Field Bio-Sample Detector Prepared by Integrated-Passive-Device Fabrication," *IEEE Access*, vol. 11, pp. 72306-72315, 2023.

[4] S. Oh, I. Hossen, J. Luglio, G. A. Justin, J.E. Richie, H. Medeiros,and C.H. Lee, "On-site/in situ continuous detecting ppb-Level metal ions in drinking water using block loop-gap resonators and machine learning," *IEEE Trans. on Instrumentation and Measurement*, vol. 70, pp. 1-9, 2021.

[5] K. Zhang, R.K. Amineh, Z. Dong, and D. Nadler, "Microwave sensing of water quality," *IEEE Access*, vol. 7, pp. 69481-69493, 2019.

[6] R.K. Amineh, M. Ravan and D. Tandel, "Detection of Water Pollutants With a Nonuniform Array of Microwave Sensors," *IEEE Trans. on Instrumentation and Measurement*, vol. 72, pp. 111, 2023.

[7] C.G. Juan, E. Bronchalo, B. Potelon, C. Quendo, E. Avila-Navarro, and J.M. Sabater-Navarro, "Concentration measurement of microliter volume water–glucose solutions using Q factor of microwave sensors," *IEEE Trans. on Instrumentation and Measurement*, vol. 68, pp. 2621-2634, Jul. 2019.

[8] S. Harnsoongnoen, "Metamaterial-inspired microwave sensor for detecting the concentration of mixed phosphate and nitrate in water," *IEEE Trans. on Instrumentation and Measurement*, vol. 70, pp. 1–6, 2021.

[9] Y. SK, O. NTJ, L. SCJ, M.D. NS, D. SH, A. AYI, and S. CF, "Microwave sensing of ammonia and iron concentration in water based on complementary double split-ring resonator," *Sensors and Actuators Reports*, vol. 3, 2021.

[10] scikit-learn, "Machine Learning in Python - Support Vector Machines," [Online]. Available: *https://scikit-learn.org/stable/modules/svm.html*, 2024.

[11] R. David, J. Duke, A. Jain, V.J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T Wang, and P. Warden, "TensorFlow-lite micro: embedded machine learning on tinyml systems," *MLSys Conf.*, 2021.

[12] Edgeimpulse, "Edge Impulse Processing Blocks," [Online]. Available: *https://github.com/edgeimpulse/processing-blocks/tree/master/spectral_analysis*, 2023.

[13] STMicroelectronics, "STM32L073RZ - Ultra-low-power Arm Cortex-M0+ MCU with 192 Kbytes of Flash memory, 32 MHz CPU, USB, LCD," [Online]. Available: *https://www.st.com/en/microcontrollers-microprocessors/stm32l073rz.html*, 2023.