Joint Learning and Channel Coding for Error-Tolerant IoT Systems based on Machine Learning

Xiaochen Tang, *Member, IEEE*, Pedro Reviriego, *Senior Member, IEEE*, Wei Tang, *Member, IEEE*, David G. M. Mitchell, *Senior Member, IEEE*, Fabrizio Lombardi, *Fellow, IEEE* and Shanshan Liu, *Member, IEEE*

***Abstract*—In several machine learning (ML)-based Internet of Things (IoT) systems, data is captured by IoT devices and then transmitted over a wireless channel for remote processing. Since noise often appears on the channel (so causing data corruption and consequently an incorrect ML result), channel protection must be provided to guarantee an acceptable error rate for the transmitted data, especially in safety-critical applications. An often-used protection technique employs error correction codes (ECCs); however, even with some improved designs, the power dissipation required by an ECC implementation may still not meet the strict requirements of hardware-constrained platforms. To address this issue, a "joint learning and channel coding" (JLCC) scheme is proposed in this paper. In such a scheme, the ML model is retrained using two methods to tolerate some channel errors, such that the system requires an ECC with significantly lower protection capability. Since ML training is executed remotely, JLCC achieves a significant power reduction for ECC without introducing any additional overhead to the IoT device. An electrocardiogram (ECG) system is taken as a case study to illustrate the proposed JLCC scheme and evaluate its effectiveness. A low-density parity-check (LDPC) code is employed for protection of the system with/without JLCC; its analysis and implementation are presented. Simulation results show that, when employing JLCC with the proposed two retraining methods, an average reduction of 29.15% and 34.82% in the dissipated power is achieved for the ECG sensor when compared to the original system.**

***Impact Statement*—Internet of Things (IoT) are one of the most promising technology frameworks in today's digital world; the use of machine learning (ML) is important for improving the efficiency of analyzing data transmitted between different IoT devices. However, recent research has demonstrated that the data transmitted in an ML-based IoT system can be affected by errors in the wireless channel, causing an incorrect outcome of the entire system; this may lead to a catastrophic consequence in safety-critical applications, so error protection for data transmission must be performed to guarantee a reliable result. The joint learning and channel coding (JLCC) technique proposed in this paper overcomes the limitation of existing error protection approaches in terms of hardware overhead (for metrics such as power dissipation). Therefore, it is very attractive for ML-based IoT systems implemented in hardware-constrained platforms for a wide variety of safety-critical applications like smart healthcare and transportation.**

***Index Terms*—Machine learning, channel coding, error-tolerance, IoT system, electrocardiogram, low-density parity-check codes.**

Manuscript received October 11, 2022, revised December 2, 2022, and accepted January 4, 2022. The work was supported by the U.S. Department of Defense under Contract W52P1J2093009, by NSF under Grant 1652944, Grant 2015573, Grant 1953961, Grant 1812467, Grant 2145917, Grant 2148358, Grant 1757207, and by the Spanish Agencia Estatal de Investigation under Grant PID2019-104207RB-I00 and Grant TSI-063000-2021-127 (*Corresponding author: Shanshan Liu*).

Xiaochen Tang, Wei Tang, David G. M. Mitchell and Shanshan Liu are with the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM 88003, USA (email: {hypnus, wtang, dgmm, ssliu}@nmsu.edu).

Pedro Reviriego is with the Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, 28040 Madrid, Spain (email: pedro.reviriego@upm.es).

Fabrizio Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02215, USA (email: lombardi@ece.neu.edu).

I. Introduction

THE Internet of Things (IoT) combined with advances in machine learning (ML) techniques is driving the development of innovative artificial intelligence based systems for different applications, including the safety-critical domains such as healthcare [1], [2], transportation [3], driverless vehicles [4], finance [5], and defense [6]. In these systems, IoT devices are often utilized to only capture sensor data (e.g., a wearable biosensor); the data is then transmitted over a wireless channel and remotely processed by edge or cloud platforms using software-based ML algorithms. Error-tolerance is usually provided in ML-based IoT systems for reliable operation in safety-critical applications [7]; otherwise, errors causing data corruption may have catastrophic consequences (e.g., potential life and property loss). Therefore, protection techniques and strategies targeting different parts of the system, such as the model computation [8], data transmission [9], memories [10], ML algorithms [11] and error detection schemes [12], have been pursued in the technical literature.

As one of the critical components in an IoT system, wireless communication is prone to be affected by different types of channel noise that can cause erroneous received data bits [13]. The often-used solution is to employ a channel coding scheme using error correction codes (ECCs) [14], [15]. In such a scheme, some parity bits are generated in the encoding process (prior to transmission) and then for example padded to the data being protected. If channel errors affect either the original data being protected, or the parity bits when transmitted, then the redundancy can be employed to attain an acceptable bit error

rate (BER) [13] (or symbol error rate (SER) for ternary data [9]) in the decoding process once the data is received.

Therefore, when employing channel coding, additional hardware must be utilized for the encoding/decoding process as well as increasing the number of transmitted parity bits, resulting in an increase of power/energy; this overhead is significant when an ECC is utilized for error correction of multiple bits. As IoT devices in many applications face stringent requirements, there is a need to reduce the power dissipated by the ECC encoding and data transmission (decoding complexity is not a primary issue of concern in this scenario because it is often performed remotely by a host station), while of course ensuring that errors do not compromise system performance. This poses a challenge for ECC design in IoT systems because conventional coding schemes incur a significant overhead to attain a low error rate for the decoded data. Therefore, improvements in ECC design with low hardware overhead have received significant attention in the technical literature [9], [16].

For example, ternary low-density parity-check (LDPC) codes [9] have been recently proposed to provide channel protection for an electrocardiogram (ECG) monitoring and arrhythmia classification system [17] (which is introduced in Section II in more detail). LDPC codes are one of the most widely used ECC types and have been used in various communication standards (e.g., 5G new radio [18], IEEE 802.11n for wireless local area network (WLAN) [19], IEEE 802.16e for mobile broadband wireless access system [20], DVB-S2/T2/C2 for digital video broadcasting [21]). However, LDPC codes with a conventional construction may not be applicable to IoT systems due to the high hardware complexity. Therefore, alternative designs, such as that proposed for the ECG system of [17] must be considered to meet the hardware requirements. In this application, the IoT device is a wearable biosensor that converts the input signals to ternary symbol streams for efficient transmission, so a corresponding ternary LDPC code with a low complexity encoder is designed [9]. Compared to traditional LDPC codes with binary bits, this ternary coding scheme provides hardware-efficient encoding and thus it significantly reduces power dissipation. However, this scheme still must transmit a considerable number of parity bits, which increases the power required for data transmission. Therefore, a more efficient protection scheme, as will be proposed in is paper, for these IoT systems is of interest. Although we consider an ECG monitoring system as a case study, the proposed techniques are widely applicable to general low power IoT sensing applications.

ML algorithms are known to be resilient to not-critical errors (e.g., causing only a small deviation from the exact data) [22]; hence this promises the use of less complex coding schemes with a lower protection capability required for channel protection in ML-based IoT systems. This occurs because protection only needs to ensure that there is little or no error on the *final outcome* of the ML models (i.e., not for the absence of errors on the received sensor data). Therefore, it is of interest to take this consideration one step further by training the ML models with inputs affected by the expected error rate. In this case, the system learns to tolerate these errors and requires less protection (even no protection in some cases) for data transmission.

Training ML models with noisy data has been shown to improve performance because it can mitigate overfitting [23] or improve the model robustness[1] when performing regression tasks [24]. However, to the best of the authors' knowledge, training ML classifiers to tolerate transmission/channel errors for reducing protection overhead has not been considered in the technical literature. In this paper, we pursue the study of such a joint scheme and propose a so-called joint learning and channel coding (JLCC) technique. The main contributions of this paper are as follows:

- To show that ML models can be trained to tolerate channel errors (due to transmission) on data;
- To show that a joint design of ML and channel coding significantly reduces the protection overhead against channel errors;
- To demonstrate the benefits of the proposed JLCC technique in an IoT biomedical application (i.e., an ECG system).

The rest of this paper is organized as follows. Section II reviews the ECG system (used as a case study to analyze the proposed scheme) and the associated error protection scheme using ternary LDPC codes. Section III presents the proposed JLCC scheme and two retraining methods applied to it. The JLCC scheme and its required ternary LDPC design are implemented in Section IV; its effectiveness is also evaluated in terms of the protection overhead introduced by the ternary LDPC implementation and compared to the traditional scheme. Finally, the paper ends with the conclusion in Section V.

II. Preliminaries

A. ECG Monitoring and Arrythmia Classification Systems

An ECG monitoring and arrhythmia classification system is a type of ML-based IoT system for smart medical applications [25]. A recently presented ECG scheme was shown to be an attractive candidate for next generation wearable ECG sensor systems due to its advantages in terms of low power implementation and continuous real-time ECG monitoring [17]. In such a system, as shown in Fig. 1, a wireless sensor (made of a ternary delta modulator) and a radio frequency (RF) data transmitter capture, process, and then transmit human ECG information. Specifically, the ternary delta modulator converts the captured analog ECG data into ternary symbol streams consisting of "1", "0", and "-1" symbols that are transmitted over the wireless channel. Once the ECG data arrives to the receiver of a remote station, a machine learning algorithm starts to analyze them and performs the arrhythmia classification; if a dangerous/abnormal type of heartbeats (e.g., supraventricular ectopic beat (SVEB) or ventricular ectopic beat (VEB) [17]) is found, an alert that triggers human diagnosis is generated for

[1] Robustness refers to the capability of tolerating errors [24]; this is also applicable in this paper.

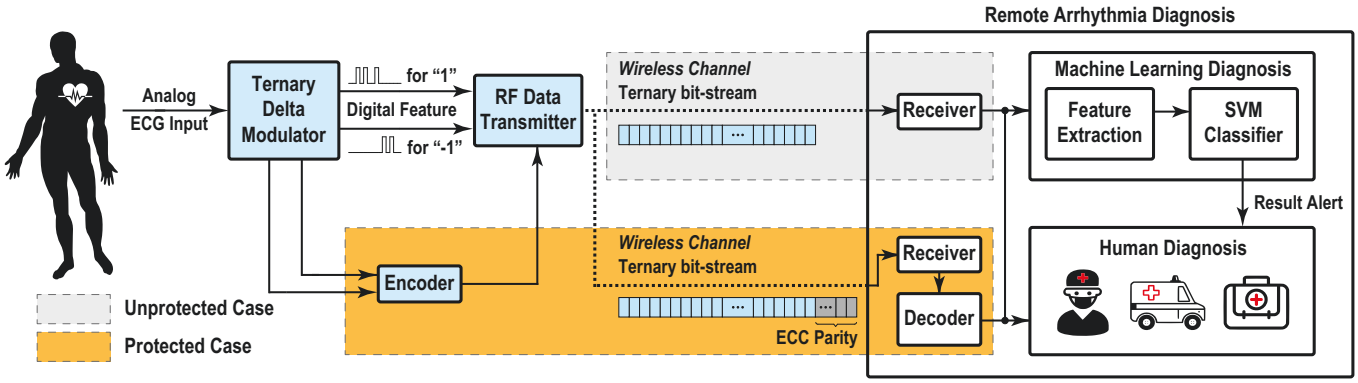

Fig. 1. ECG system: a wireless wearable ECG sensor with remote machine learning-based arrhythmia diagnosis.

further treatment. This ECG system is considered as a case study in this paper to analyze and evaluate the proposed scheme.

Due to its low computational complexity but yet with a high learning performance, a linear-kernel based support vector machine (SVM) has been proposed in [26] to perform classification in an ECG system. It will also be considered in the case study of this paper to investigate the proposed retraining methods. An SVM model is typically trained by initially solving (1) to obtain the non-negative value of the Lagrange multiplier a associated to each of the N training samples [27], where x and y denotes the feature and class of the training samples respectively, i.e.,

$$max\left(\sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j x_i x_j\right) | \sum_{i=1}^{N} a_i y_i = 0. \quad (1)$$

Once this is completed, the support vectors that belong to the training dataset, are determined to establish the decision margin and boundary of an SVM for classification. Finally, the SVM model is obtained based on the weight w and bias b that are calculated by $w = \sum_{i=1}^{N} a_i y_i x_i$ and $b = \frac{1}{N_{SV}}\sum_{i\in SV}\left(t_i - \sum_{j\in SV} a_j t_j x_i^T x_j\right)$, where N_{SV} denotes the number of support vectors (SV), t denotes the target value corresponding to each training sample and $t \in \{-1,1\}$. To classify a sample X during inference using a linear-kernel function (binary classification is considered in this manuscript), its class Y is predicted as

$$Y(X) = sign\left(\sum_{i=1}^{f}(w_i X + b_i)\right), \quad (2)$$

where f is the number of features of X; the positive and negative signs refer to the different classes respectively.

To achieve a high classification accuracy in the ECG system, the SVM model proposed in [26] utilizes two such models with a constant combination coefficient. Its detailed design can be found in [26] and will not be presented in this paper.

B. Impact of Channel Errors on Classification

Next, the impact of errors caused by channel errors during data transmission is initially evaluated to validate the necessity of employing a protection scheme in the ECG system. A dataset (#207 patient) of the widely used benchmark MIT-BIH arrhythmia database [28], including the heartbeat data of 47 patients and each with a few thousand heartbeat signals, is used

[2] Since the impact on VEB classification is negligible (found in our simulations), then only the SVEB classification is considered in this paper.

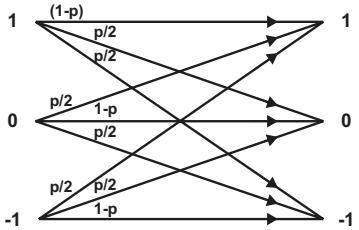

Fig. 2. Ternary symmetric channel model with error probability p [9].

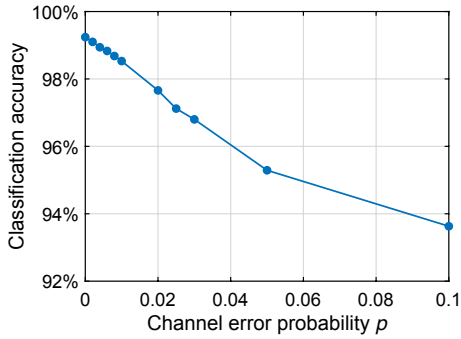

Fig. 3. ECG classification accuracy under channel errors at different probability.

to show the impact of channel errors on the ECG classification accuracy[2]. To model the channel errors, the widely used ternary symmetric channel model given in Fig. 2 with error probability p is considered in this paper for error injection.[3] As per the sampling rate of the ECG sensor, a stream consisting of 500 data points (ternary symbols) that reflects a critical waveform (e.g., a QRS complex) of heartbeats, is transmitted [9]. Therefore, for different p, errors are randomly injected in each transmitted 500 data points; this is then analyzed by the SVM classifier to perform the SVEB classification.

Fig. 3 shows the average results of the classification accuracy by repeating the simulation over 1000 times for different cases (this number of simulation runs is selected because the results are consistent with 10000 simulations); it is observed that the accuracy of 99.24% in the error-free case decreases to 93.63% when p increases to 0.1. Such accuracy degradation is not acceptable in practical applications, especially when the ECG system is applied to some patients with a severe heart disease (i.e., a correct monitor and classification of heartbeat signals is essential). Therefore, error protection must be performed to

[3] The evaluation approach can also be extended to other types of channel models, which do not change the qualitative conclusions drawn in this paper.

handle channel errors. The protection technique is reviewed in the next subsection.

C. Ternary LDPC Codes

As outlined previously, channel coding is typically utilized to protect the transmitted data against noise-induced errors [14]. When employing a systematic ECC, the data being protected is encoded by generating parity bits prior to transmission; following this, the codeword consisting of the original data being protected and its parity bits is transmitted. Consider Fig. 1 again, it illustrates an example of an ECC that is employed to protect the transmitted data in the ECG system. If channel errors affect any bits of the codeword during transmission, they are remotely handled in the decoding process. Finally, an outcome that is error-free, or with an acceptable SER (related to an acceptable signal to noise ratio of a given application), is generated.

However, these codes often incur in a significant overhead when implemented in hardware; so, a hardware efficient ECC design for data communication has been advocated as introduced previously. When ECCs are employed in an IoT system, the incurred overhead mainly includes two parts: i) additional data transmission power due to the ECC parity bits (which must be transmitted along with the data); ii) additional circuit area and power incurred by the encoder (that is typically integrated in the IoT devices). The decoding overhead is usually not an issue in an IoT system, because the decoder is implemented in the remote station that does not have strict requirements on hardware overhead.

Recently, a class of ternary LDPC codes with low encoding complexity [9] have been proposed to provide error correction for a ternary symbol stream in the ECG system introduced above. Such codes are constructed by forming the parity-check matrix $\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2]$ with a quasi-cyclic structure based on the following two sub-matrices

$$\mathbf{H_1} = \begin{bmatrix} \delta_{0,0}\mathbf{P}_s^{b_{0,0}} & \delta_{0,1}\mathbf{P}_s^{b_{0,1}} & \cdots & \delta_{0,m-1}\mathbf{P}_s^{b_{0,m-1}} \\ \delta_{1,0}\mathbf{P}_s^{b_{1,0}} & \delta_{1,1}\mathbf{P}_s^{b_{1,1}} & \cdots & \delta_{1,m-1}\mathbf{P}_s^{b_{1,m-1}} \\ \vdots & \vdots & & \vdots \\ \delta_{k-1,0}\mathbf{P}_s^{b_{k-1,0}} & \delta_{k-1,1}\mathbf{P}_s^{b_{k-1,1}} & \cdots & \delta_{k-1,m-1}\mathbf{P}_s^{b_{k-1,m-1}} \end{bmatrix}, \quad (3)$$

$$\mathbf{H_2} = \begin{bmatrix} \mathbf{I}_s & \mathbf{0}_s & \cdots & \cdots & \gamma_{k-1}\widehat{\mathbf{P}_s^1} \\ \gamma_0\mathbf{I}_s & \mathbf{I}_s & \mathbf{0}_s & \cdots & \mathbf{0}_s \\ \mathbf{0}_s & \gamma_1\mathbf{I}_s & \mathbf{I}_s & \cdots & \mathbf{0}_s \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0}_s & \cdots & \mathbf{0}_s & \gamma_{k-2}\mathbf{I}_s & \mathbf{I}_s \end{bmatrix}. \quad (4)$$

In these two sub-matrices, $\delta \in GF(3) = \{0, 1, 2\}$, $\gamma \in GF(3)\backslash\{0\}$, $\mathbf{P}_s$ is a $s \times s$ circulant permutation matrix constructed by circularly shifting the rows of the $s \times s$ identity matrix $\mathbf{I}_s$ with $b \in \{1, 2, \ldots, s\}$ positions to the left. $\widehat{\mathbf{P}_s^1}$ is given by (5) and is constructed by removing the right corner "1" in $\mathbf{P}_s^1$, i.e.,

$$\widehat{\mathbf{P}_s^1} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}. \quad (5)$$

With the parity-check matrix $\mathbf{H}$, the code requires $K = k \cdot s$ ternary parity symbols to protect $M = m \cdot s$ data symbols. This results in a code rate of $M/(M + K)$. When employing these ternary LDPC codes in the ECG system, the ternary data "-1" can be considered as "2" in the encoding process; moreover, since the ECG sensor operates at a low sampling rate and is hardware constrained, an efficient encoding approach is also designed in [9] to meet these requirements. As per the quasi-cyclic structure of $\mathbf{H}$, the encoding can be designed to execute on a serial fashion (which is acceptable considering the sampling rate of ECG data). Consider an error-free codeword of the ternary LDPC code $\mathbf{v} = [d_0, d_1, \ldots, d_{M-1}, p_0, p_1, \ldots, p_{K-1}]$ (which consists of M data symbols d and K parity symbols p) and the parity check rule $\mathbf{v} \cdot \mathbf{H}^T = 0$ [14]. (6) is obtained because the first row of $\mathbf{H}_2$, given by (4), has only the first element as "1". Therefore, the first parity symbol p_0 can be calculated as

$$\delta_{0,0} \cdot d_{s-b_{0,0}} + \delta_{0,1} \cdot d_{2s-b_{0,1}} + \cdots + \delta_{0,m-1} \cdot d_{k \cdot s - b_{0,m-1}} + p_0 \equiv 0, \quad (6)$$

where all operations are computed over $GF(3)$. Once p_0 is generated, p_s can be subsequently obtained in the same way by considering the $(s+1)^{\text{th}}$ row, because all elements in this row are "0"s except the first and the $(s+1)^{\text{th}}$ elements, i.e., only p_s is unknown in the following expression

$$\delta_{1,0} \cdot d_{s-b_{1,0}} + \delta_{1,1} \cdot d_{2s-b_{1,1}} + \cdots + \delta_{1,m-1} \cdot d_{k \cdot s - b_{1,m-1}} + \gamma_0 p_0 + p_s \equiv 0. \quad (7)$$

By repeating this procedure as per the quasi-cyclic structure of $\mathbf{H}$, all parity symbols can be generated serially to complete encoding (i.e., establishing a relationship similar to (6) and (7) each time, in which only one parity symbol is calculated).[4]

However, even though this encoding process enables a small encoder circuit implemented in the ECG sensor, the ternary LDPC codes still require a considerable number of parity bits (e.g., the same as the number of ECG symbols being protected if the code rate is 1/2) to achieve small error rates [9]; this incurs in a significant power dissipation for data transmission, because all required parity bits need to be transmitted together with the original data. Therefore, a scheme that can further reduce the LDPC protection overhead by achieving higher code rates while still meeting the required protection is necessary when an IoT system is implemented on power-constrained platforms.

III. Proposed JLCC Scheme

In this section, a method referred to as joint learning and channel coding (JLCC) is proposed to improve the error-tolerance of ML-based IoT systems with low hardware overhead. The details of JLCC are first presented; then, two retraining methods are discussed as employed in such a scheme.

A. Joint Learning and Channel Coding

The proposed JLCC scheme relies on making the ML algorithm robust by learning channel errors, so that the burden

[4] For more details of the encoding/decoding process of this ternary LDPC codes, please refer to [9].

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2023.3235778

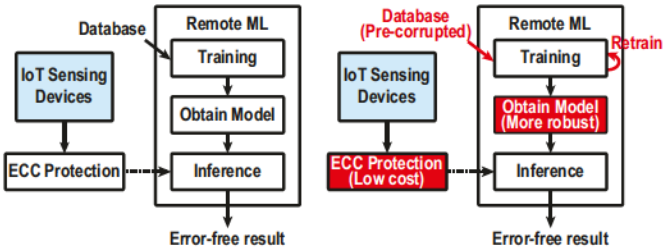

Fig. 4. ML-based IoT system: (a) traditional scheme without JLCC; (b) proposed scheme with JLCC (differences from (a) are marked in red).

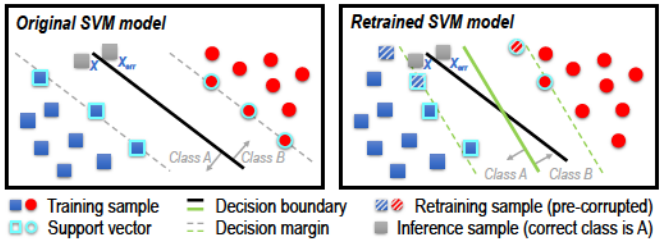

Fig. 5 Examples of SVM inference for binary classification: sample X_{err} that contains channel errors is incorrectly (correctly) classified as Class B (A) by the original (retrained) model.

of error protection for the wireless channel can be reduced. The framework of an ML-based IoT system employing the JLCC scheme is illustrated in Fig. 4. In the traditional scheme (Fig. 4(a)), the IoT biosensor device captures data and converts it to a digital bit-stream that is protected by ECC when transmitting in the wireless channel; once data is obtained by the remote ML part, the model trained by using an existing database performs inference on the received data to make a classification. This predicted classification result is error-free (or with an acceptable accuracy loss due to errors), because an ECC has been employed for error-tolerance by introducing additional hardware. When utilizing JLCC (Fig. 4(b)), the result is still error-free, but the ECC protection overhead is reduced. This is achieved by using a database that is pre-corrupted with errors (related to the channel properties) to retrain the ML model.

Consider the SVM model used in an ECG system as an example. As shown in Fig. 5, an error-free sample X can be correctly classified as Class A by both models. However, its classification result changes to B if X has been corrupted by channel errors during transmission (denoted as X_{err}) in the traditional scheme, i.e., $Y(X_{err}) \neq Y(X)$. This issue is addressed in the proposed scheme; in JLCC, the ML model is retrained by introducing additional C pre-corrupted samples (that present the features of channel errors) to solve

$$max\left(\sum_{i=1}^{N+C} a_i - \frac{1}{2}\sum_{i=1}^{N+C}\sum_{j=1}^{N+C} a_i a_j y_i y_j x_i x_j\right) | \sum_{i=1}^{N+C} a_i y_i = 0. \quad (8)$$

Therefore, the SVM model (i.e., the decision boundary) is modified based on the retrained support vectors as shown in Fig. 5; this achieves the following

$$\begin{aligned} Y(X_{err}) &= sign\left(\sum_{i=1}^{f}(w_i' X' + b_i')\right) \\ &= sign\left(\sum_{i=1}^{f}(w_i' X + b_i')\right) = Y(X), \end{aligned} \quad (9)$$

where w' and b' are the weight and bias values based on the retrained support vectors. Therefore, X_{err} will also be classified as Class A by using the retrained model.

By introducing a number of sufficient pre-corrupted samples to retrain, the obtained model can tolerate at least some channel errors on the received samples during inference. Therefore, compared to the traditional scheme, an ECC with a lower protection capability (or even no ECC) requiring less additional hardware can be used in the proposed scheme to handle the remaining errors and provide an error-free result. Note that even though the original training dataset may also include some noise (caused for example by an inexact feature extraction), this noise has different properties from the channel errors; this makes the original model less robust than the retrained model in the presence of channel errors.

Since the ML algorithm executes remotely, JLCC does not introduce any overhead during data sensing or data transmission. Even though in some applications in which the ML is implemented remotely on a hardware-constrained platform (such as an embedded system or edge device), only inference (e.g., an ML accelerator) is implemented, while training is executed off-line; thus, JLCC also incurs in no overhead for the hardware platform because it only focuses on training.

The process described above implies that the most critical step of JLCC is the retraining process. Since the ML model is modified after retraining, it is important to ensure that the retrained model does not degrade the classification performance in the error-free case when compared to the original model. This can be achieved by using the proposed retraining methods discussed in the next subsection. The effectiveness of the proposed JLCC scheme is evaluated in Section IV by considering the ECG system as a case study.

B. Retraining Method

Fig. 6 illustrates the proposed retraining schemes for JLCC. Consider a given channel error probability p, the original training dataset is first injected by random errors with a probability p_{tr} for generating a noisy/corrupted copy; both the original and the noisy datasets are then combined and used to retrain the model. In particular, two retraining methods are considered:

1) ***Method 1***: generate the noisy dataset by injecting errors with $p_{tr} = p$; this is performed to let the model learn and tolerate errors with the given channel error probability.
2) ***Method 2***: generate the noisy dataset by injecting errors with $p_{tr} \leq p$, i.e., allowing all possible errors with a several pre-defined probability values up to the given value. By doing this, the model learns more types of errors (i.e., those with different probabilities) and thus is expected to be more robust.

These two initial methods are investigated in this paper to show the effectiveness of the proposed JLCC scheme; however, other extended/improved solutions (e.g., with $p_{tr} \geq p$) that may achieve a better training performance could be explored. This is left for future work.

Next, in both cases, the ML model is retrained by using the same method as for training the original model. Once an initially retrained model is obtained, inference is executed to

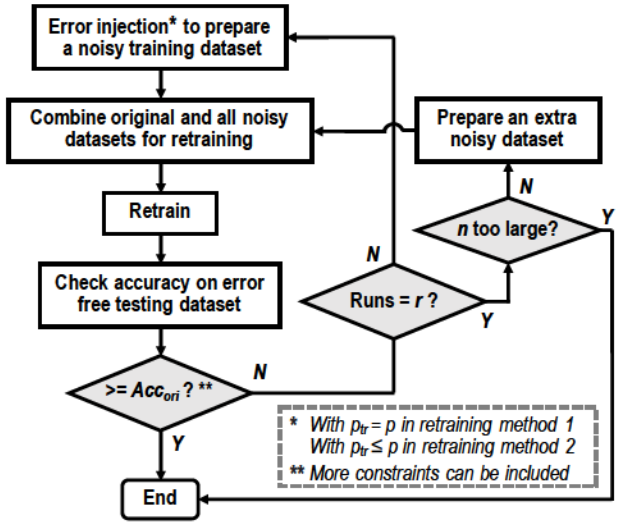

Fig. 6. Retraining Process (Acc_{ori} is the classification accuracy of the original model in the error-free case, r is the number of repeated retraining runs, and n is the size of the noisy training dataset).

check its classification accuracy on the original error-free testing dataset; if the result is the same or better than the error-free accuracy of the original model (denoted to as Acc_{ori}), the expected retrained model is obtained, and retraining completes. Note that the reasons for probably achieving a better accuracy than Acc_{ori} in the retrained model are (i) the increased size of the training dataset; (ii) the introduction of noisy data, which mitigates overfitting. However, since the pre-corrupted samples added in the proposed retraining method present the properties of channel errors that usually have a limited error probability, they are likely to be similar to the error-free samples; hence, the retrained model is not expected to provide a major improvement in accuracy in the error-free case (as verified by the evaluation results presented in Section IV). To obtain a retrained model for other figures of merit, more constraints can be included (i.e., in addition to the comparison with Acc_{ori}) in the last step of retraining; this is discussed in detail in Section IV.

If the initially retrained model is not acceptable (i.e., its classification accuracy is smaller than Acc_{ori} in the error-free case), the entire process as discussed above is repeated until the expected model is found. Since the erroneous data samples in the noisy training dataset are randomly generated, repeating the process is likely to generate a model that learns about the errors and then can tolerate similar ones when p is small. For a larger value of p, more data samples are corrupted and thus the number of considered error patterns increases. As a consequence, this makes the model incapable to learn the errors using only a small noisy training dataset. Therefore, if the retraining process is repeated r times but a good model is not obtained yet, an additional noisy dataset (using the same error injection method as in the first step) is generated and both noisy datasets participate in retraining. This is illustrated in Fig. 6. Let the size of the noisy dataset be n times (i.e., n = 1, 2, ...) the size of the original error-free dataset. If n is too large, then the complexity of training is unacceptable for the remote ML station and the proposed retraining process ends/fails. However, this case is unlikely to occur because, in practice, the value of p is limited.

TABLE I
DESCRIPTION OF DIFFERENT DATASETS

Dataset*	Number of data samples	Classification accuracy
#207	2704	99.24%
#209	3009	97.85%
#232	1786	96.50%

*Since the classification of SVEB and not-SVEB is considered in this paper, few datasets from the MIT-BIH database [28] with a larger number of SVEB data samples are selected.

Compared to the first retraining scheme, the second scheme tends to require a larger value of n but it generates a more robust model because more error patterns are taken into account. Note that the size of the retraining dataset used to achieve the given constraints has no impact on the hardware overhead of the wireless devices, because retraining is performed off-line.

IV. EVALUATION

A. Setup

The ECG system introduced in Section II-A is taken as a case study to evaluate the effectiveness of the proposed JLCC scheme. The same SVM model of [26] is used as the ML classifier. A few datasets of the MIT-BIH arrhythmia database [28], including patient #207, patient #209, and patient #232, are used to check the impact of channel errors on the ML results (i.e., the classification accuracy in this case study). Table I shows the description of each dataset and their classification accuracy of the original ML model (pretrained in [26]) in the error-free case. To also assess the proposed JLCC scheme in terms of hardware efficiency at an acceptable error-tolerance, the ECG system protected using the ternary LDPC code from [9] is evaluated and compared as a baseline.

The error injection process is conducted by using the same method as the one described in Section II-B and the ternary symmetric channel model of Fig. 2. The averaged SVEB classification accuracy among 1000 simulations is evaluated in the presence of errors, each with error probability p (varying from 0.002 to 0.100). For the parameters r (the number of repeated retraining runs) and n (the size of noisy training dataset) used in the retraining process, exemplary values of r = 1000 and n_{max} = 200 are considered. Next, the robustness of the retrained model obtained using the proposed JLCC scheme is evaluated; then the associated LDPC codes are implemented and compared with the baseline in terms of hardware overhead.

B. Robustness of Retrained Model

As illustrated in Fig. 6, to achieve no degradation of performance in the retrained model, retraining is performed by increasing the value of n (for each n, the simulation is repeated r = 1000 times) and completes once a model with the same or larger classification accuracy compared to the error-free result given in Table I is found. Table II shows the value of n required for the considered datasets using different retraining methods. As expected, the value of n increases for a larger error probability in both retraining schemes because the learning is harder for the model in the presence of a larger number of the errors. Hence, in such cases, more training data samples are required to provide a better classification performance. This is

TABLE II
REQUIRED SIZE OF NOISE TRAINING DATA N UNDER ERRORS FOR DIFFERENT P

Channel error probability p			0.002	0.004	0.006	0.008	0.010	0.020	0.025	0.030	0.050	0.100
Required n	#207	Method 1	1	1	1	2	2	4	40	60	100	200
		Method 2	1	2	2	3	3	5	50	80	120	200
	#209	Method 1	1	1	1	2	2	3	40	60	90	170
		Method 2	1	1	1	2	2	4	50	70	120	200
	#232	Method 1	1	1	1	1	2	2	40	60	90	160
		Method 2	1	1	1	2	2	2	40	70	120	190

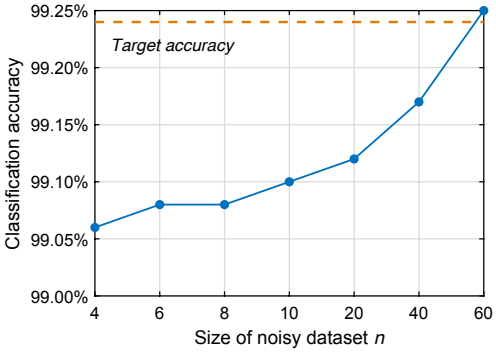

Fig. 7. Classification accuracy of the retrained model for dataset #207 at different sizes of noisy retraining data (under $p = 0.03$).

verified in the example plotted in Fig. 7 for dataset #207 and $p = 0.03$, where a higher classification accuracy is achieved when the size of the retraining dataset is increased.

As seen from Table II, the increase of n tends to be exponential with p, especially for values of p larger than 0.02. This is likely because the number of error patterns increases exponentially, and a corresponding larger number of data samples are required for learning. This also applies to the difference between the results using different retraining methods. In method 2, errors with different probabilities ($\leq p$) are injected in the training dataset and consequently the number of realized error patterns is larger than for method 1 (that considers errors with only the fixed probability p). Therefore, as shown in Table II, the value of n for method 2 is equal to or larger than for method 1 in all cases; moreover, the difference of n tends to become larger for a larger value of p, because in such a case, the model retrained using method 2 must learn a larger number of error patterns.

A second experiment performed is to evaluate the robustness of the retrained models in the presence of errors. Once the retrained models are obtained, their classification accuracy on the testing dataset injected by errors with different p is checked. In each case, the accuracy of the original model is also evaluated for comparison. Table III reports the results for dataset #207. Since it is of interest to show the robustness of a model retrained by targeting p with fewer errors (i.e., the occurrence of errors with probability p is considered as the worst case of the channel), the accuracy of the retrained models for an error probability smaller than p is also checked. As per Table III, the classification accuracy results are shown to have the following trends:

- In the error-free case, the retrained models using both proposed methods maintain the same or a slightly higher classification accuracy compared to the original model. This is expected because at least the same accuracy (as for the original model) is the constraint when retraining. A possible reason for obtaining a slightly higher accuracy seems to be that a scheme by which training ML models introduces noise to the input variables is

TABLE III
CLASSIFICATION ACCURACY OF DIFFERENT MODELS FOR DATASET #207

Classifier model		Classification accuracy on testing data with different p										
		Error-free	0.002	0.004	0.006	0.008	0.010	0.020	0.025	0.030	0.050	0.100
Original of [26]		99.24%	99.10%	98.94%	98.83%	98.68%	98.53%	97.66%	97.12%	96.80%	95.29%	93.63%
Retrained with*	$p_{tr} = 0.002$	99.24%	99.12%	-	-	-	-	-	-	-	-	-
	$p_{tr} \leq 0.002$	99.24%	99.12%	-	-	-	-	-	-	-	-	-
	$p_{tr} = 0.004$	99.24%	99.14%	99.05%	-	-	-	-	-	-	-	-
	$p_{tr} \leq 0.004$	99.24%	99.21%	99.18%	-	-	-	-	-	-	-	-
	$p_{tr} = 0.006$	99.24%	99.23%	99.21%	99.13%	-	-	-	-	-	-	-
	$p_{tr} \leq 0.006$	99.24%	99.22%	99.21%	99.15%	-	-	-	-	-	-	-
	$p_{tr} = 0.008$	99.24%	99.24%	99.20%	99.13%	99.11%	-	-	-	-	-	-
	$p_{tr} \leq 0.008$	99.24%	99.24%	99.23%	99.20%	99.17%	-	-	-	-	-	-
	$p_{tr} = 0.010$	99.24%	99.23%	99.20%	99.18%	99.10%	99.02%	-	-	-	-	-
	$p_{tr} \leq 0.010$	99.24%	99.24%	99.24%	99.21%	99.18%	99.11%	-	-	-	-	-
	$p_{tr} = 0.020$	99.24%	99.23%	99.23%	99.23%	99.22%	99.21%	98.86%	-	-	-	-
	$p_{tr} \leq 0.020$	99.24%	99.24%	99.24%	99.24%	99.23%	99.18%	98.72%	-	-	-	-
	$p_{tr} = 0.025$	99.24%	99.25%	99.25%	99.24%	99.24%	99.24%	99.24%	99.20%	-	-	-
	$p_{tr} \leq 0.025$	99.24%	99.25%	99.25%	99.24%	99.24%	99.24%	99.24%	99.21%	-	-	-
	$p_{tr} = 0.030$	99.25%	99.25%	99.24%	99.24%	99.24%	99.24%	99.24%	99.21%	99.07%	-	-
	$p_{tr} \leq 0.030$	99.25%	99.25%	99.25%	99.24%	99.24%	99.24%	99.24%	99.22%	99.09%	-	-
	$p_{tr} = 0.050$	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.11%	98.93%	-
	$p_{tr} \leq 0.050$	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.12%	99.01%	-
	$p_{tr} = 0.100$	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.18%	99.09%	98.89%
	$p_{tr} \leq 0.100$	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.24%	99.22%	99.15%	98.92%

*$p_{tr} = p$ refers to retraining method 1 and $p_{tr} \leq p$ refers to retraining method 2.

TABLE IV
CLASSIFICATION ACCURACY OF DIFFERENT MODELS FOR DATASET #209

Classifier model		Classification accuracy on testing data with different p										
		Error-free	0.002	0.004	0.006	0.008	0.010	0.020	0.025	0.030	0.050	0.100
Original of [26]		97.85%	97.78%	97.69%	97.61%	97.52%	97.42%	96.85%	96.52%	96.24%	94.88%	91.41%
Retrained with*	$p_{tr} = 0.002$	97.85%	97.81%	-	-	-	-	-	-	-	-	-
	$p_{tr} \le 0.002$	97.85%	97.81%	-	-	-	-	-	-	-	-	-
	$p_{tr} = 0.004$	97.85%	97.82%	97.76%	-	-	-	-	-	-	-	-
	$p_{tr} \le 0.004$	97.85%	97.83%	97.76%	-	-	-	-	-	-	-	-
	$p_{tr} = 0.006$	97.85%	97.83%	97.75%	97.65%	-	-	-	-	-	-	-
	$p_{tr} \le 0.006$	97.85%	97.85%	97.82%	97.76%	-	-	-	-	-	-	-
	$p_{tr} = 0.008$	97.85%	97.83%	97.82%	97.72%	97.65%	-	-	-	-	-	-
	$p_{tr} \le 0.008$	97.85%	97.85%	97.82%	97.78%	97.73%	-	-	-	-	-	-
	$p_{tr} = 0.010$	97.85%	97.85%	97.83%	97.76%	97.69%	97.59%	-	-	-	-	-
	$p_{tr} \le 0.010$	97.85%	97.85%	97.83%	97.76%	97.69%	97.64%	-	-	-	-	-
	$p_{tr} = 0.020$	97.85%	97.85%	97.83%	97.82%	97.79%	97.77%	97.66%	-	-	-	-
	$p_{tr} \le 0.020$	97.85%	97.85%	97.85%	97.83%	97.82%	97.81%	97.66%	-	-	-	-
	$p_{tr} = 0.025$	97.85%	97.85%	97.85%	97.83%	97.79%	97.77%	97.74%	97.68%	-	-	-
	$p_{tr} \le 0.025$	97.85%	97.85%	97.85%	97.84%	97.79%	97.77%	97.76%	97.70%	-	-	-
	$p_{tr} = 0.030$	97.85%	97.85%	97.85%	97.83%	97.82%	97.80%	97.78%	97.72%	97.62%	-	-
	$p_{tr} \le 0.030$	97.85%	97.85%	97.85%	97.85%	97.83%	97.80%	97.78%	97.73%	97.64%	-	-
	$p_{tr} = 0.050$	97.85%	97.85%	97.85%	97.85%	97.85%	97.80%	97.78%	97.75%	97.66%	97.62%	-
	$p_{tr} \le 0.050$	97.85%	97.85%	97.85%	97.85%	97.85%	97.83%	97.80%	97.78%	97.73%	97.62%	-
	$p_{tr} = 0.100$	97.85%	97.85%	97.85%	97.85%	97.85%	97.85%	97.85%	97.82%	97.77%	97.70%	97.60%
	$p_{tr} \le 0.100$	97.85%	97.85%	97.85%	97.85%	97.85%	97.85%	97.85%	97.83%	97.80%	97.76%	97.59%

*$p_{tr} = p$ refers to retraining method 1 and $p_{tr} \le p$ refers to retraining method 2.

helpful for reducing overfitting, which improves the model performance (as indicated in [23]);

- In the presence of errors, the retrained models using both proposed methods achieve a higher accuracy compared to the original model in all cases. Moreover, the second retraining method (i.e., $p_{tr} \le p$) performs better because it enables the model to learn a larger number of error patterns as discussed previously;
- In both schemes, the model retrained with a larger error probability tends to be more robust (i.e., achieving a higher accuracy). This is observed in each column of Table III; for example, when p = 0.002, the model retrained with p_{tr} = 0.006 offers a higher accuracy (99.23%) than the model retrained with p_{tr} = 0.004 (99.14%), followed by the one using p_{tr} = 0.002 (99.12%). This again is expected because the model retrained targeting a larger p learns more errors, so achieves a better error-tolerance.

These trends are also the case for the results of the other two datasets (#209 and #232) reported in Tables IV and V.

Next, the case for dataset #207 is taken as an example to discuss the protection technique in more detail (i.e., the ternary LDPC codes) required in two different scenarios:

- The first scenario considers an unprotected system that can tolerate a given SER, for which the loss in ML accuracy for this SER is acceptable at system level;

TABLE V
CLASSIFICATION ACCURACY OF DIFFERENT MODELS FOR DATASET #232

Classifier model		Classification accuracy on testing data with different p										
		Error-free	0.002	0.004	0.006	0.008	0.010	0.020	0.025	0.030	0.050	0.100
Original of [26]		96.50%	96.08%	95.64%	95.21%	94.88%	94.60%	93.08%	92.36%	91.75%	89.59%	86.36%
Retrained with *	$p_{tr} = 0.002$	96.50%	96.21%	-	-	-	-	-	-	-	-	-
	$p_{tr} \le 0.002$	96.50%	96.21%	-	-	-	-	-	-	-	-	-
	$p_{tr} = 0.004$	96.50%	96.21%	96.13%	-	-	-	-	-	-	-	-
	$p_{tr} \le 0.004$	96.50%	96.21%	96.15%	-	-	-	-	-	-	-	-
	$p_{tr} = 0.006$	96.50%	96.23%	96.17%	95.78%	-	-	-	-	-	-	-
	$p_{tr} \le 0.006$	96.50%	96.24%	96.19%	95.88%	-	-	-	-	-	-	-
	$p_{tr} = 0.008$	96.50%	96.26%	96.15%	95.87%	95.62%	-	-	-	-	-	-
	$p_{tr} \le 0.008$	96.50%	96.30%	96.19%	95.93%	95.77%	-	-	-	-	-	-
	$p_{tr} = 0.010$	96.50%	96.34%	96.15%	95.90%	95.56%	95.27%	-	-	-	-	-
	$p_{tr} \le 0.010$	96.50%	96.38%	96.20%	95.93%	95.80%	95.29%	-	-	-	-	-
	$p_{tr} = 0.020$	96.50%	96.36%	96.19%	95.96%	95.77%	95.24%	95.10%	-	-	-	-
	$p_{tr} \le 0.020$	96.50%	96.40%	96.22%	96.05%	95.85%	95.38%	95.19%	-	-	-	-
	$p_{tr} = 0.025$	96.50%	96.41%	96.24%	96.05%	95.89%	95.42%	95.17%	94.77%	-	-	-
	$p_{tr} \le 0.025$	96.50%	96.44%	96.26%	96.09%	95.90%	95.45%	95.19%	94.79%	-	-	-
	$p_{tr} = 0.030$	96.50%	96.41%	96.27%	96.15%	96.00%	95.56%	95.30%	95.18%	94.62%	-	-
	$p_{tr} \le 0.030$	96.50%	96.41%	96.27%	96.18%	96.01%	95.45%	95.32%	95.20%	94.67%	-	-
	$p_{tr} = 0.050$	96.50%	96.45%	96.32%	96.30%	96.11%	95.79%	95.45%	95.25%	95.12%	94.60%	-
	$p_{tr} \le 0.050$	96.50%	96.47%	96.35%	96.30%	96.15%	95.83%	95.50%	95.29%	95.16%	94.62%	-
	$p_{tr} = 0.100$	96.50%	96.50%	96.44%	96.33%	96.24%	95.97%	95.69%	95.36%	95.20%	94.88%	94.57%
	$p_{tr} \le 0.100$	96.50%	96.50%	96.47%	96.35%	96.29%	96.03%	95.77%	95.42%	95.21%	94.91%	94.60%

*$p_{tr} = p$ refers to retraining method 1 and $p_{tr} \le p$ refers to retraining method 2.

TABLE VI
PROTECTION OF TERNARY LDPC CODES REQUIRED FOR MODELS WITH A GIVEN SER_{TARGET} OR AD_{TARGET}

Channel error probability	Protection capability of ternary LDPC codes required for different models*								
	With an SER_{target} of 0.002			With an SER_{target} of 0.004			With an AD_{target} of 0.5%		
	Original model of [26]	Retrained model 1 of JLCC	Retrained model 2 of JLCC	Original model of [26]	Retrained model 1 of JLCC	Retrained model 2 of JLCC	Original model of [26]	Retrained model 1 of JLCC	Retrained model 2 of JLCC
$p = 0.002$	No need	No need	No need	No need	No need	No need	No need	No need	No need
$p = 0.004$	SER = 0.002	SER = 0.002	No need	No need	No need	No need	No need	No need	No need
$p = 0.006$	SER = 0.002	No need	No need	SER = 0.004	No need	No need	No need	No need	No need
$p = 0.008$	SER = 0.002	No need	No need	SER = 0.004	No need	No need	SER = 0.006	No need	No need
$p = 0.010$	SER = 0.002	SER = 0.008	No need	SER = 0.004	No need	No need	SER = 0.006	No need	No need
$p = 0.020$	SER = 0.002	SER = 0.010	SER = 0.010	SER = 0.004	SER = 0.010	No need	SER = 0.006	No need	No need
$p = 0.025$	SER = 0.002	No need	No need	SER = 0.004	No need	No need	SER = 0.006	No need	No need
$p = 0.030$	SER = 0.002	SER = 0.025	SER = 0.025	SER = 0.004	No need	No need	SER = 0.006	No need	No need
$p = 0.050$	SER = 0.002	SER = 0.030	SER = 0.030	SER = 0.004	SER = 0.030	No need	SER = 0.006	No need	No need
$p = 0.100$	SER = 0.002	SER = 0.030	SER = 0.050	SER = 0.004	SER = 0.050	SER = 0.050	SER = 0.006	No need	No need

* The retrained models obtained in the proposed JLCC scheme are different for each p; they are retrained using the retraining method 1 or 2 with $p_{tr} = p$ or $p_{tr} \leq p$, respectively.

- The second scenario considers a system that can tolerate a margin for degradation on the error-free ML accuracy.

In both cases, the goal is to achieve a target ML accuracy for the entire system that is specified as (i) a target SER (SER_{target}) in the unprotected system, or (ii) a margin over the error-free ML accuracy that is denoted as a target accuracy degradation (AD_{target}). These two error-tolerant metrics define the limitation of performance degradation that the IoT application permits. The values of SER_{target} or AD_{target} considered next are just examples, which do not affect the qualitative conclusion; they can be determined as per the requirement of practical applications.

***Achieving an SER**$_{target}$*: this scenario refers to the case in which the system tolerates an SER that is equal to SER_{target} when not protected; moreover, the SER_{target} can be converted to a target ML performance of the unprotected system under channel errors with $p = SER_{target}$. For example, in the ECG system, an SER_{target} of 0.002 refers to a classification accuracy of 99.10% in the original unprotected model (shown in Table III); this value indicates that the models must achieve such an accuracy in the presence of errors. Therefore, in this scenario, ternary LDPC codes with different protection capabilities are employed to protect different models (including the original model of [26], and the retrained models using methods 1 and 2) to achieve a satisfactory classification accuracy ($\geq$ 99.10%).

Table VI (left part) gives the ternary LDPC code required for different models to meet $SER_{target} = 0.002$. When the channel error probability p is 0.002, no protection is required for all models, because it is equal to SER_{target} (i.e., these errors can be accepted). When p increases, the original model must be protected by employing a ternary LDPC code that achieves an SER of 0.002. This is required because, as shown in Table III, errors degrade the model accuracy to less than 99.10% (that is related to the SER_{target} of 0.002 as discussed previously). However, for the retrained models obtained in the proposed JLCC scheme, ternary LDPC codes with a larger SER than for the original model are allowed; moreover, there is no need to employ LDPC coding in some cases. For example, when p = 0.010, the original model requires a ternary LDPC code with SER = 0.002 to achieve the target accuracy of 99.10%, while the retrained model 1 (with $p_{tr} = 0.010$) requires a ternary LDPC code with SER = 0.008, and the retrained model 2 (with $p_{tr} \leq$ 0.010) does not require any protection. This occurs because, as shown in Table III, the retrained model 1 (2) has a classification accuracy of 99.10% (99.11%) under errors with p = 0.008 (0.010). Hence, these two models are more robust, so require less or no protection.

Next, consider a larger value of SER_{target}, such as 0.004 that refers to a target classification accuracy of 98.94% as per Table III (i.e., more errors can be accepted). The required ternary LDPC code for different cases is also given in Table VI (middle part). Compared to the results for SER_{target} of 0.002, the required LDPC code tends to have a larger SER (less protection), and no protection is required in more cases; this is as expected, because the requirement of error-tolerance in the system is reduced.

***Achieving an AD**$_{target}$*: this scenario refers to the case when the IoT application accepts a specific degradation AD_{target} in the error-free classification accuracy, such as 0.5% (i.e., one additional incorrect result is allowed among 1000 classifications). In this case, the SVM classifier that has an accuracy of 99.24% in the error-free case must achieve an accuracy of at least 98.74% in the presence of errors. As per the model robustness shown in Table III, the ternary LDPC code required for different models is also given and compared in Table VI (right part). Only the original model requires the protection of the LDPC code with an SER of 0.006 when the error probability is equal to or larger than 0.008. This occurs because the model still provides an accuracy of at least 98.83% for errors with a smaller probability. For the retrained models, no protection is required in all cases because their accuracy results are better than 98.74% under all errors.

As expected, when considering the robustness of a model under errors with different probabilities, it should become weaker (or at least remain the same) under more errors, and thus requires more powerful/stronger (or at least the same) protection. However, this is only the case for the original model as shown in Table VI, but not for both retrained models. For example, the results for retrained model 1 with $p_{tr} = p$ of 0.004, 0.010, and 0.020 seem to intuitively be "No need" (because a protection can even be not needed for larger p). As introduced

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2023.3235778

TABLE VII
DIFFERENT MODELS RETRAINED WITH DIFFERENT CONSTRAINTS FOR THE #207 DATASET

Classifier model*		Classification accuracy on testing data with different p							Required n	Required ternary LDPC when $p_{tr} = p$
		Error-free	0.002	0.004	0.006	0.008	0.010	0.020		
$P_{tr} = 0.004$	Retrained model 1	99.24%	99.14%	99.05%	-	-	-	-	1	SER = 0.002
	Retrained model 1'	99.24%	99.15%	99.11%	-	-	-	-	1	No need
$P_{tr} = 0.010$	Retrained model 1	99.24%	99.23%	99.20%	99.18%	99.10%	99.02%	-	2	SER = 0.008
	Retrained model 1'	99.24%	99.23%	99.23%	99.21%	99.13%	99.10%	-	2	No need
$P_{tr} = 0.020$	Retrained model 1	99.24%	99.23%	99.23%	99.23%	99.22%	99.21%	98.86%	4	SER = 0.010
	Retrained model 1'	99.24%	99.23%	99.23%	99.23%	99.23%	99.23%	99.10%	6	No need

* Retrained model 1 is obtained with the initial constraint of achieving an accuracy larger than or equal to Acc_{ori} in the error-free case, while retrained model 1' is obtained with an additional constraint of achieving an accuracy larger than or equal to 99.10% under errors.

previously (Fig. 6), the retrained model is obtained once its accuracy in the error-free case is equal to or larger than the original model (i.e., with such a single constraint) and a different model can be obtained if the retraining process is executed again. Therefore, under errors with a given p, the model retrained with a larger p_{tr} may not always be more robust than the one with a smaller p_{tr}.

As explained in the discussion of Fig. 6, additional constraints can be included during the retraining process. Therefore, a third investigation is pursued to retrain some models again by introducing more constraints to show that they may offer a higher robustness. The three retrained models mentioned above (i.e., using method 1 with $p_{tr} = p$ of 0.004, 0.010, and 0.020, respectively) and the case of a target SER of 0.002 are considered; the goal is to retrain them again to achieve the target SER with no LDPC code protection (i.e., their results are expected as "No need" in Table VI (left)). Hence, in addition to the initial constraint, the constraint of achieving an accuracy larger than or equal to 99.10% under errors (which relates to the $SER_{target} = 0.002$ as discussed previously) is also included in the retraining process.

Table VII shows the classification accuracy of the new retrained models (named as retrained model 1') for dataset #207 under different errors and their required protection scheme. For comparison, some results for the initial version of these three models (named as retrained model 1) previously reported in Tables II, III, and VI are given again in this table. More robust models are obtained in all cases when considering two constrains using the same or larger size of noisy training data (i.e., n).

Overall, in the proposed JLCC scheme, the retrained model can tolerate some errors, so the required protection capability of the ternary LDPC code is lower than for the original model. This typically leads to a higher code rate (i.e., smaller number of parity bits to be transmitted). The hardware overhead required for protection is evaluated in the next subsection.

C. LDPC Hardware Overhead

As per the design method of [9], ternary LDPC codes with three code rates (including R = 2/3, 1/2, and 1/3) are designed to protect the ECG data made of 500 symbols per transmission. It should be noted that these codes are just taken as examples to illustrate the effectiveness of the proposed scheme; any additional improvement for a better decoding process is beyond the scope of this paper and is left for future work. Fig. 8 plots the error correction capability for different cases. The smallest

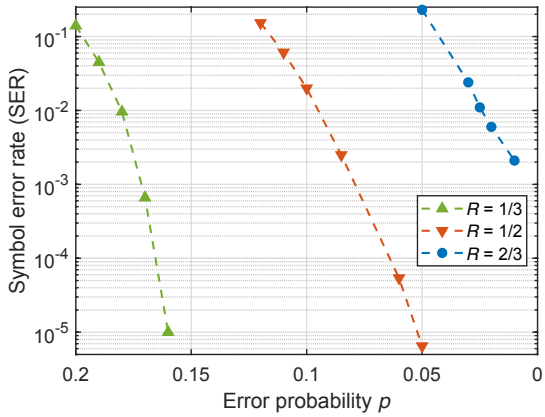

Fig. 8. Error correction of ternary LDPC code at different code rate in the presence of channel errors for different probability.

code rate (i.e., R = 1/3) provides the highest protection as expected, because the corresponding code possesses the largest proportion of parity symbols that also need to be transmitted. Considering both Tables VI and Fig. 8, ternary LDPC codes with the largest code rate that can achieve SER_{target} or AD_{target} for different error probabilities, can be employed for different models to provide sufficient protection while incur smallest overhead. For example, when $SER_{target} = 0.002$ and $p = 0.030$, the original model of [26] requires a code with R = 1/2, while a code with R = 2/3 is sufficient for protecting the retrained models of the proposed JLCC scheme.

As shown in Fig. 1, the hardware implementation of an ECG sensor includes a ternary Delta modulator and a RF data transmitter in the unprotected case. When using LDPC protection, an encoder is also required to generate the parity symbols and the RF data transmitter dissipates additional power to transmit them. As indicated in [9], the area overhead and latency introduced by the encoder are not an issue for the ECG system because the circuit is small, and the sensor operates at a low frequency. Therefore, only the power dissipation required for the ECG sensor in the original scheme and the proposed JLCC scheme are evaluated and compared next.

The evaluation results of our previous ECG system design show that the ternary Delta modulator dissipates a power of 302 nW at a sampling rate of 1 kHz [2] and the RF transmitter consumes an energy of 1.156 nJ for each bit [29]. Since two channels are utilized to transmit the ternary symbols, the entire data transmission power is 2.312 μW (i.e., 1.156 nJ × 2 × 1 kHz) in the unprotected case; due to the transmitted parity data, this number increases to 3.468 μW, 4.624 μW, and 6.936 μW

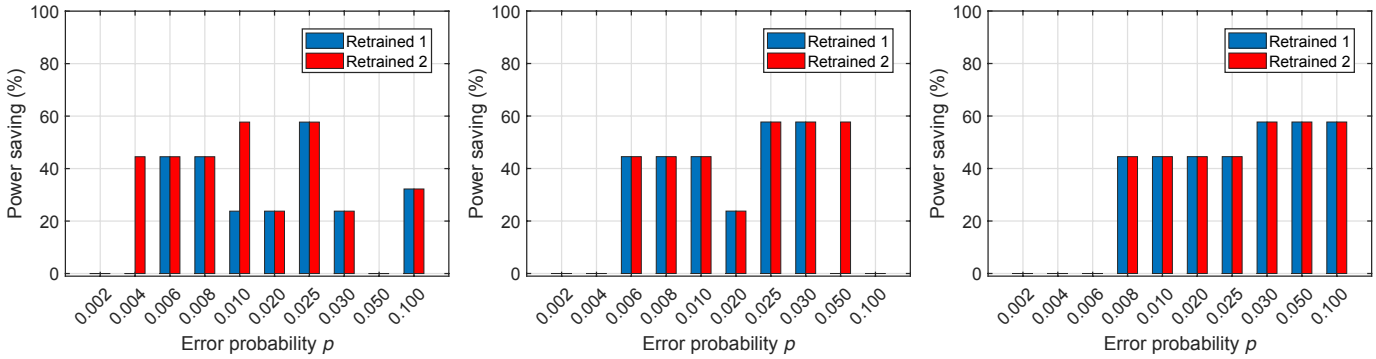

Fig. 9. Power saving of the ternary ECG sensor achieved by the retrained models of the proposed JLCC scheme compared to the original model of [26] for dataset #207: (a) results for $SER_{target} = 0.002$; (b) results for $SER_{target} = 0.004$; (c) results for $AD_{target} = 0.5\%$.

TABLE VIII
EVALUATED POWER RESULTS OF THE ECG SENSOR IN DIFFERENT SCHEMES

Scheme		Power Dissipation (μW)				
		Ternary Delta modulator	RF data transmitter		Encoder	Entire ECG sensor
			Original data	Parity data		
Unprotected		0.302	2.312	0	0	2.614
Protected by ternary LDPC	$R = 2/3$	0.302	2.312	1.156	0.942	4.712
	$R = 1/2$	0.302	2.312	2.312	1.256	6.182
	$R = 1/3$	0.302	2.312	4.624	1.884	9.122

when using the ternary LDPC codes with a code rate of 2/3, 1/2, and 1/3, respectively. The encoders for the different codes are implemented using VHDL and the designs are synthesized using Synopsys Design Compiler by mapping to a 65 nm technology library at a frequency of 2 kHz (which is sufficiently high to support the sensor). Table VIII gives the synthesis results of different encoders as well as the power results of other components of the ECG sensor. The ternary LDPC protection with a larger code rate is shown to incur a significantly lower power dissipation for error protection and for the entire ECG sensor. This indicates that the proposed JLCC scheme can achieve a power reduction compared to the original scheme. Again, in the example when $SER_{target} = 0.002$ and $p = 0.030$, the protection power dissipation for the retrained models (using a code with $R = 2/3$) is 2.098 μW while that for the original model (using a code with $R = 1/2$) is 3.568 μW; this leads to a power reduction of 41.20% for error protection and 23.78% for the entire system (as per Table VIII).

To show the advantage of the proposed JLCC scheme in terms of hardware overhead, the power dissipation of the entire ECG sensor is utilized as evaluation metric (which is more reasonable than considering the protection power only); dataset #207 is again considered as an example to compare the power results of different schemes. Fig. 9 plots the power saving of the ECG sensor achieved by the retrained models of the proposed JLCC scheme compared to the original model of [26]. A reduction of more than 40% is shown in many cases, because no protection is required by the retrained models as per Table VI. Moreover, an average saving of 29.15% is achieved by the retained model 1 and 34.82% is achieved by the retrained model 2; this is also expected because the retrained model 2 with more constraints during retraining is more robust as discussed previously. Fig. 9 also shows no power saving for some error probabilities; this occurs because, when p is very small, both the original and retrained models do not need error protection (i.e., the systems dissipate the same power). This can also happen in more cases when SER_{target}/AD_{target} is larger, as verified in Fig. 9. For the results when $p = 0.050$ in Fig. 9(a) or $p = 0.100$ in Fig. 9(b), no power saving is shown due to the limited choice of the ternary LDPC codes considered in this paper; here, even though the retrained models for these error probabilities allow a lower protection capability (shown in Table VI), they need to use the same code as in the original model (with $R = 1/2$ as per Fig. 8).

Again, note that the proposed JLCC scheme only needs to remotely retrain the ML model, which is often executed off-line and prior to applying the entire system for classification. Hence, the significant power saved by the proposed JLCC scheme is achieved without introducing any additional hardware overhead on the wireless devices (e.g., the ECG sensor).

V. CONCLUSION

In this paper, a joint learning and channel coding (JLCC) scheme has been proposed to address the issue of power dissipation posed by error correction codes (ECCs) when they are employed for channel protection in ML-based IoT systems. By remotely retraining the ML model to learn errors using two proposed methods in the JLCC scheme, the requirement of ECC protection and the overhead introduced in the IoT devices have been considerably reduced.

An existing electrocardiogram (ECG) monitoring and arrhythmia classification system protected using ternary low-density parity-check (LDPC) codes has been considered as an example as application of JLCC. Its performance has also been evaluated (note that JLCC is also applicable to other systems with other ECC techniques). When targeting different error-tolerant scenarios for the ECG system, the LDPC codes required in different schemes to achieve the same ML classification performance have been analyzed and implemented. The simulation results show that by employing JLCC, an average power saving of 29.15% and 34.82% of the ECG sensor can be achieved by using the proposed two retraining methods when compared to the original system. Therefore, the proposed JLCC scheme is very attractive for

ML-based IoT systems that are implemented in hardware-constrained platforms and used in safety-critical applications.

REFERENCES

[1] S. Tiwari, P. Chanak and S. K. Singh, "A review of the machine learning algorithms for Covid-19 case analysis," *IEEE Transactions on Artificial Intelligence*, early access, Jan. 2022.

[2] M. Abubaker, "Detection of cardiovascular diseases in ECG images using machine learning and deep learning methods," *IEEE Transactions on Artificial Intelligence*, early access, Mar. 2022.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015.

[4] G. De La Torre, P. Rad, K. K. R. Choo, "Driverless vehicle security: Challenges and future research opportunities," *Future Generation Computer Systems*, vol. 108, pp. 1092-1111, Jul. 2020.

[5] S. Mishra and A. K. Tyagi, "The role of machine learning techniques in internet of things-based could applications," in *Artificial Intelligence-based Internet of Things Systems*, Springer, Cham, pp. 105-135, 2022.

[6] M. Payal, P. Dixit, T. V. Sairam, and N. Goyal, "Robotics, AI, and the IoT in defense systems," *AI and IoT-based Intelligent Automation in Robotics,* pp. 109-128, Apr. 2021.

[7] J. J. Zhang, K. Liu, F. Khalid, M. A. Hanif, S. Rehman, T. Theocharides, *et al*., "Building robust machine learning systems: Current progress, research challenges, and opportunities," in *Proceedings of the 56th Annual Design Automation Conference*, pp. 1-4, Jun. 2019.

[8] S. Liu, P. Reviriego, X. Tang, W. Tang, and F. Lombardi, "Result-based re-computation for error-tolerant classification by a support vector machine," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 62–73, Oct. 2020.

[9] Y. Liu, X. Tang, D. G. Mitchell, and W. Tang, "Ternary LDPC error correction for arrhythmia classification in wireless wearable electrocardiogram sensors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021 (Early access).

[10] V. Sharma, N. Gupta, A. P. Shah, S. K. Vishvakarma, and S. S. Chouhan, "A reliable, multi-bit error tolerant 11T SRAM memory design for wireless sensor nodes," *Analog Integrated Circuits and Signal Processing*, vol. 107, no. 2, pp. 339-352, May 2021.

[11] P. N. Whatmough, S. K. Lee, D. Brooks, and G. Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 9, pp. 2722-2731, Jun. 2018.

[12] A. Power and G. Kotonya, "Providing fault tolerance via complex event processing and machine learning for IoT systems," in *Proceedings of the 9th International Conference on the Internet of Things*, pp. 1-7, Oct. 2019.

[13] R. A. Shafik, M. S. Rahman, and A. R. Islam, "On the extended relationships among EVM, BER and SNR as performance metrics," in *IEEE International Conference on Electrical and Computer Engineering*, pp. 408-411, Dec. 2006.

[14] S. Lin and D. J. Costello, *Error control coding*. Prentice hall Scarborough, 2001.

[15] M. Bettayeb, S. Chunaim, N. Mohamed, and Q. Nasir, "Error correction codes in wireless sensor networks: A systematic literature review," in *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pp. 1-6, Mar. 2019.

[16] D. Burshtein and G. Miller, "Efficient maximum-likelihood decoding of LDPC codes over the binary erasure channel," *IEEE Transactions on Information Theory*, vol. 50, nol. 11, pp. 2837-2844, Oct. 2004.

[17] X. Tang and W. Tang, "An ECG delineation and arrhythmia classification system using slope variation measurement by ternary second order delta modulators for wearable ECG sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, no. 5, pp. 1053-1065, Sep. 2021.

[18] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28-34, Mar. 2018.

[19] "IEEE standard for information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput," *IEEE Std 802.11n-2009*, pp. 1–565, Oct. 2009.

[20] "IEEE standard for local and metropolitan area networks - Part 16: Air interface for fixed and mobile broadband wireless access systems - Amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands," *IEEE Std 802.16e-2005*, pp. 1–822, Feb. 2006.

[21] I. Eizmendi, M. Velez, D. Gómez-Barquero, J. Morgade, V. Baena-Lecuyer, M. Slimani, and J. Zoellner, "DVB-T2: The second generation of terrestrial digital video broadcasting system," *IEEE Transactions on broadcasting*, vol. 60, no. 2, pp. 258–271, Apr. 2014.

[22] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, Nov. 2017.

[23] J. Brownlee, "Train neural networks with noise to reduce overfitting," *Machine Learning Mastery*, 2019.

[24] A. Rusiecki, M. Kordos, T. Kamiński, and K. Greń, "Training neural networks on noisy data," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, pp. 131–142, Jun. 2014.

[25] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, "A survey on application of machine learning for internet of things," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 8, pp. 1399-1417, Aug. 2018.

[26] X. Tang, Z. Ma, Q. Hu, and W. Tang, "A real-time arrhythmia heartbeats classification algorithm using parallel delta modulations and rotated linear-kernel support vector machines," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 4, pp. 978–986, Jul. 2019.

[27] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. New York: Springer, 2006.

[28] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, May 2001.

[29] Q. Hu, X. Tang, and W. Tang, "Integrated asynchronous ultrawidedband impulse radio with intrinsic clock and data recovery," *IEEE Microwave and Wireless Components Letters*, vol. 27, no. 4, pp. 416-418, 2017.