Lightweight Lossy/Lossless ECG Compression for Medical IoT Systems

Yangyang Chang, *Graduate Student Member, IEEE*, and Gerald E. Sobelman, *Life Senior Member, IEEE*

***Abstract*—Monitoring patients with heart disease can be done by analyzing the electrocardiogram (ECG). However, the large amount of data poses a burden for a system that is implemented as an Internet of Things system with limited memory and computation capabilities. Traditionally, lossless compression methods have been favored to reduce the memory requirements due to the critical nature of the application. However, if the reconstruction of a lossy signal does not significantly affect diagnosis capability, then those methods may become attractive due to their larger compression ratios (CRs). In this article, we propose a hybrid lossy/lossless compression system with good signal fidelity and CR characteristics. The performance is evaluated after decompression using deep neural networks (DNNs) that have been shown to have good classification capabilities. For the Clinical Outcomes in Digital Electrocardiology (CODE) data set, the proposed hybrid compressor can achieve an average CR of 5.18 with a mean-squared error (MSE) of 0.20, and DNN-based diagnosis of the decompressed waveforms has, on average, only 0.8 additional erroneous diagnoses out of a total of 402 cases compared to using the original ECG data. For the PTB-XL data set, the hybrid compressor can achieve a high average CR of 4.91 with an MSE of 0.01. In addition, the decompressed ECGs have only a 2.46% lower macro averaged area under the receiver operating characteristic curve (AUC) score than when using the original ECGs.**

***Index Terms*—Deep neural, electrocardiogram (ECG), lossless Huffman coding, lossy compression.**

I. Introduction

CARDIOVASCULAR disease (CVD) is the leading cause of death worldwide according to the World Health Organization (WHO) [1], and electrocardiogram (ECG) analysis is a common clinical procedure to detect CVD [2]. In particular, long-term monitoring of ECGs of CVD patients can be used beneficially to alert doctors to abnormalities when they occur. Such a system requires three major components: 1) small-sized mobile patient equipment; 2) a communications network; and 3) hospital equipment to receive the data [3]. Specifically, the patient uses a portable ECG device (e.g., a handheld 12-lead electrocardiographic device [4]) in an Internet of Things (IoT)-based healthcare system to collect the information.

Manuscript received 22 April 2023; revised 27 June 2023 and 12 August 2023; accepted 12 November 2023. Date of publication 29 November 2023; date of current version 26 March 2024. *(Corresponding author: Gerald E. Sobelman.)*

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota-Twin Cities, Minneapolis, MN 55455 USA (e-mail: chan1729@umn.edu; sobelman@umn.edu).

Digital Object Identifier 10.1109/JIOT.2023.3336995

The collected ECG data is stored on the patient's own local device and can be uploaded to the hospital's diagnostic equipment or database for doctors to inspect. However, the part of the system on the patient's side would require a large amount of memory to store the long-term collection of ECG signals. Therefore, effective and reliable data compression methods are necessary for a practical implementation.

There are two types of ECG compression techniques, namely, lossy and lossless. Lossless ECG data compression is used to prevent mistakes in diagnosis but generally provides only a small compression ratio (CR) [5], [6], [7]. Lossy compression, on the other hand, permits a small data loss but can provide a larger CR [8], [9], [10], [11], [12]. Tsai and Kuo [6] demonstrated lossless compression of the lead V2 with a CR of 2.77 in an embedded platform using an ARM M4 on the 11-bit resolution MIT-BIH arrhythmia database. Using the same method, Tsai and Kuo [6] further achieved a CR of 2.89 on 12-lead ECG signals for the 16-bit PTB databases [5]. The MIT-BIH arrhythmia database has been tested by many lossless compression papers but they typically have CRs less than 4 [7], [9], [13]. It has also been tested in [8], [11], and [12] using lossy compression methods. The CR of [11] can reach 10.33 with 1.04% root mean-square difference (PRD) for all ECG recordings. Reference [10] presents results from various databases in which the ECG samples are 16-bit integers. That reference shows that different databases can lead to widely different error rates and CRs. For example, [10] can reach a 3.27 PRD and a 17.29 CR in the MIT-BIH database, but 59.79 PRD and 6.99 CR for the SVDB. Reference [9] has a CR of up to 90 on the 12-bit MIT-BIH AF database. Reference [14] provides a hybrid method in which lossless compression is applied on the main pulse regions in the waveform, known as the QRS complexes, and lossy compression is used on the other segments. Though these lossy papers can achieve a very good compression, they are difficult to implement in an automatic detection assistance application. First, these lossy approaches often focus on the compression of one or two channels of ECG recordings. The information among multiple channels can provide more useful information for the detection of complex ECG patterns. Second, the detection of the QRS complex is not sufficient to make many CVD judgments. Third, some of the methods require a large amount of computation. For example, [11] uses the combination of DWT, quantizer, and SPIHT encoder, which is a heavy burden for an embedded system.

This article proposes a lightweight hybrid ECG compressor for an IoT medical system. The hybrid compressor has a

lossy part based on an optimization method and a lossless part based on the Huffman entropy encoding method. In the proposed system, the ECG data will be compressed by a sequence of two encoders, namely, a lossy encoder followed by a lossless encoder. The lossy encoder requires only a small amount of memory and it has a small computational load consisting of just additions and multiplications. The fractional part of the normalized outputs of the lossy encoder are further compressed in the lossless encoder using Huffman coding. At the receiving side, the compressed ECG data will be decompressed by passing through a lossless decoder and then a lossy decoder. The quality of the system is evaluated by using deep neural networks (DNNs) to determine if the decompressed data can still be classified correctly. The 1st data set used [15] consists of tracings from 827 distinct patients, and the 2nd one is the PTB-XL data set [16], which consists of ECGs from 18 869 patients. The novel evaluation procedure we adopt is based on diagnoses made by DNNs on the original waveforms and on the decompressed waveforms. DNNs have achieved remarkable successes in tasks, such as healthcare applications [15], image classification [17], and speech translation [18]. Reference [15] demonstrates that the DNN outperforms cardiology residents in recognizing six types of abnormalities in 12-lead ECG recordings, with F1 scores above 80% and specificity values of more than 99%. Reference [19] indicates that the convolutional neural network (CNN) with an entropy method can achieve 89.2%, 76.5%, and 69.8% for classification of 2, 5, and 20 classes, respectively, on the PTB-XL database. In our application, we wish to know if a DNN can correctly diagnose the decompressed ECGs to the same degree of accuracy as for the original ECG signals. In this way, it will be clear whether the compression process is adversely impacting the information content in the ECGs that is important for diagnosis.

DNNs [15], [20], [21], [22], [23] which have been shown to have accurate diagnosis capabilities are used to classify the ECGs before and after decompression. For the 1st data set, the proposed hybrid compression model has an average CR of 5.18 and a mean-squared error (MSE) of 0.20, and using the decompressed signals produces only an average of 0.8 additional classification errors out of a total of 402 cases compared to the original classifier for the detection of six abnormal types of ECGs. For the 2nd data set, the proposed model achieves an average CR of 4.91 with an MSE of 0.01, and the average scores of the DNN diagnoses systems have only a 2.46% lower macro averaged area under the receiver operating characteristic curve (AUC) score than when using the original ECGs. In addition, for the 1st data set, which has a 400-Hz sampling rate, the proposed compression scheme has a small 15.19-MB total memory requirement with 3.24 million parameters in its lossy model. The lossy compression part requires only 4.12 MB of memory and 0.88 million parameters. For the 2nd data set, which has a 100-Hz sampling rate, the compression portion of the lossy model has a small 3.71-MB total memory requirement with 0.79 million parameters. The lossy compression part requires only 1.01 MB of memory with 0.22 million parameters. The small memory requirements of the compression system enable our model to be deployed in a memory- and computation-limited embedded hardware platform.

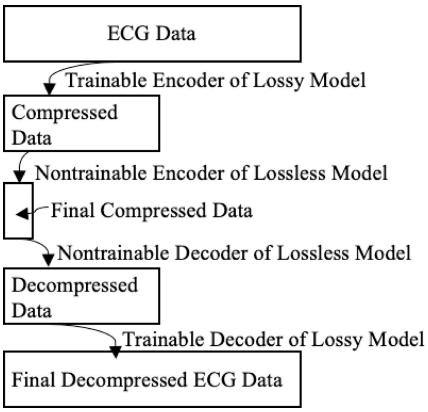

Fig. 1. Architecture and data flow of the proposed hybrid compression and decompression model. The data is compressed first using a lossy model and then by a lossless model. The decompression process applies the inverse operations in the reverse order.

II. Proposed Hybrid Compression Model

The overall architecture and the data flow of the hybrid compression method are shown in Fig. 1. The ECG signals will be first compressed by the lightweight lossy model and then compressed by the lossless coding method.

A. Proposed Lightweight Lossy Model

The lossy model is composed of three parts: 1) the encoder; 2) the decoder; and 3) the final output layer. The design concept for the lossy compression comes from information crossover between individuals in an evolutionary algorithm [24], [25], [26]. The architecture of the lossy model is shown in Figs. 2 and 3.

The architecture of the encoder has Y compression layers, where the number of layers is determined by the CR using the following relationship: $\mathrm{CR} = 2^Y$. In a compression layer, each sampled ECG signal is viewed as one individual. The length of an individual is determined by the number of channels. For example, if we have 4000 ECG samples with 12 channels, there are 4000 individuals, and each individual has 12 values. In the process of compression or decompression, the number of individuals will change but not the length of individuals. The detailed computation is summarized in Algorithm 1. Here, *Inputs* is the original ECG signal after the preprocessing; $W^E_{i,j}$ is the (i, j)th trainable weight vector to adjust the jth individual in the ith compression layer, the output Z^E is the final compressed ECG signal and the operator $\circ$ represents the Hadamard product. In each compression layer, every individual is adjusted by the corresponding weights having the same length (Algorithm 1, line 5). Then, the first half and the second half of all individuals are added to generate new individuals (Algorithm 1, line 8). After each compression layer, the total number of new individuals will be half of the number before that layer. The output Z^E will be further compressed by the lossless method, which is presented in the following section.

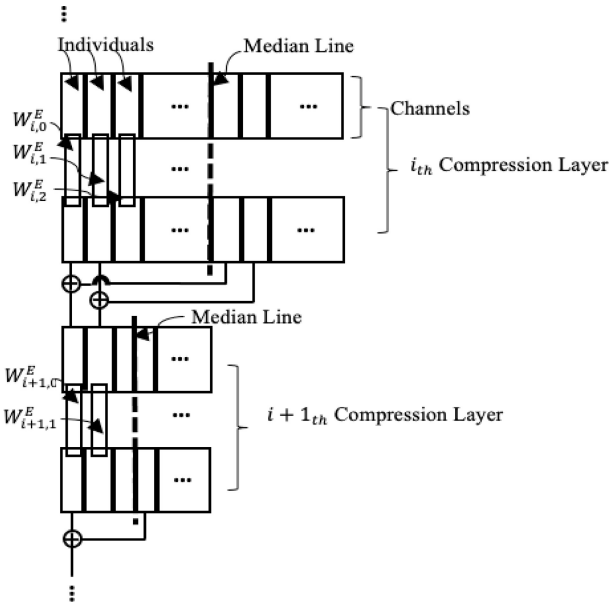

Fig. 2. Architecture of the encoder of the lossy model. It consists of a sequence of compression layers, each of which reduces the data size.

Algorithm 1 Encoder

1: Inputs: *Inputs*, Y
2: Initial $Z^E = Inputs$
3: **Loop** i from 0 to Y:
4: **Loop** j from 0 to (size(Z^E) -1):
5: $individual_j = individual_j \circ W^E_{i,j}$
6: Store $W^E_{i,j}$
7: **End Loop**
8: Z^E = (first half of individuals in Z^E) + (second half of individuals in Z^E)
9: **End Loop**
10: Output: Z^E

The architecture of the decoder and the final output layer is shown in Fig. 3. Note that the inputs to the decoder will have been already decompressed according to the lossless method. The decoder has the same number Y of layers as the encoder. Each individual has the same length as the individual at the output Z^E of Algorithm 1. For each decompression layer, the number of individuals M and the output Z^D follows $M = Z^D 2^Y$, and every individual is adjusted by the corresponding weights with the same length. The detailed process of the decoder is summarized in Algorithm 2. Here, $\text{individual}^a_{\text{jv}}$ is the jth individual of B^a_v (e.g., the ath branch of the vth group), $W^{\text{Da}}_{i,j,v}$ is the jth individual of ath branches of the vth group in the ith decompression layer, $\circ$ is the Hadamard product, and O^a is the ath branch output. The computation in each decompression layer can be summarized in the following four steps.

1) The individuals in each layer are formed into N groups (Algorithm 2, line 3), where each group has two branches. The individuals are adjusted by the weights of the first branch in forward order (Algorithm 2, line 5) but are adjusted by the second branch in reverse order (Algorithm 2, line 9). All groups undergo the same operations, namely, to multiply the individuals by the weights (Algorithm 2, lines 6 and 10).
2) The branches of all groups are combined together to form two outputs (Algorithm 2, lines 14 and 15). For example, if there are three groups, the first output is the summation of the 1st branch of the 1st group, the 2nd branch of the 2nd group, and the 1st branch of the 3rd group. The second output is the summation of the 2nd branch of the 1st group, the 1st branch of the 2nd group, and the 2nd branch of the 3rd group.
3) A set of twice as many individuals is generated by concatenating the two outputs (Algorithm 2, line 16).
4) Group normalization [27] is applied to the concatenated output (Algorithm 2, line 17). The group normalization together with the normalization in the preprocessing function can improve the precision of the decompressed data.

Algorithm 2 Decoder and Final Output Layer

1: Inputs: Z^D, Y
2: **Loop** i from 0 to Y:
3: **Loop** v from 0 to N:
4: Create memory spaces for B^0_v and B^1_v
5: (B^0_v) **Loop** j from 0 to (size(Z^D) -1):
6: $individual^0_{j,v}$= $individual_j$ of $Z \circ W^{D0}_{i,j,v}$
7: Store $W^{D0}_{i,j,v}$
8: **End Loop**
9: (B^1_v) **Loop** j from (size(Z^D) -1) to 0:
10: $individual^1_{j,v}$= $individual_j$ of $Z \circ W^{D1}_{i,j,v}$
11: Store $W^{D1}_{i,j,v}$
12: **End Loop**
13: **End Loop**
14: $O^0 = B^0_{v=0} + B^1_{v=1} + \ldots + B^D_v$(***D*** is odd if ***V*** is odd; ***D*** is even if ***V*** is even)
15: $O^1 = B^1_{v=0} + B^0_{v=1} + \ldots + B^D_v$(***D*** is even if ***V*** is odd; ***D*** is odd if ***V*** is even)
16: $Z^D = concatnate(O^0\, O^1)$ to double the number of individuals.
17: Z^D= GroupNormalization(Z^D)
18: **End Loop**
19: **Loop** j from 0 to (size(Z^{LD}) -1):
20: $individual_j = individual_j$ of $Z \circ W^F_j$
21: Store W^F_j
22: **EndLoop**
23: Output Z^D

In the final output layer, each individual from the output of the decoder is modified by the weights to generate the final ECG signals (Algorithm 2, line 20). Specifically, W^F_j is the jth weight of the jth individual in the final output layer.

B. Lossless Coding Model

The lossless compression method uses Huffman coding [28], although other types of lossless compression (e.g., Golomb–Rice or Variable Byte coding [29]) could alternatively be used. The output of the encoder Z^E is directly

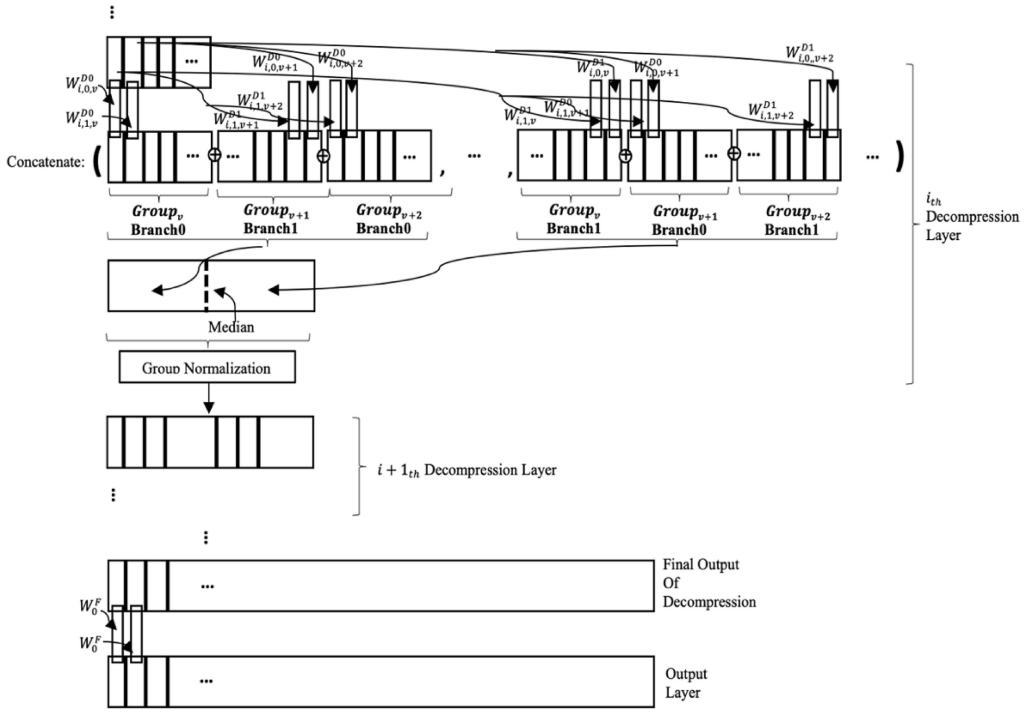

Fig. 3. Architecture of the decoder of the lossy model. It consists of a sequence of decompression layers followed by a final output layer.

input to the lossless model. The details can be summarized as follows.

1) The output of the encoder is normalized to the interval [0, 1] by $Z^E = (Z^E - \min(Z^E)/(\max(Z^E) - \min(Z^E))$. The $\min(Z^E)$ and $\max(Z^E)$ values are stored so as to be used again when performing the decompression.
2) Except for the value 1, the integer parts of values are all 0. Therefore, only the fractional part is compressed by the Huffman coding.
3) The decompression process is completed in two steps. First, the decompressed values before normalization are the integer 0 plus the decompressed fractional parts. Second, these decompressed values are mapped from the [0, 1] range back to the original range using the stored $\min(Z^E)$ and $\max(Z^E)$ values.

III. Data Sets and Processing

A. Data Sets

Two data sets were tested. The first one is the Clinical Outcomes in Digital Electrocardiology (CODE) study [15], which has a sampling rate of 400 Hz. This data set consists of 827 tracings annotated by cardiologists [15]. There are six abnormal ECG patterns included in that data set, namely, 1st degree AV block, right bundle branch block, left bundle branch block, sinus bradycardia, atrial fibrillation, and sinus tachycardia. The second data set is PTB-XL [16] consisting of 21 799 12-lead ECGs from 18 869 patients, with a sampling rate of 100 Hz.

B. Preprocessing

ECG samples from different patients or ECG machines may have various ranges and scales. Therefore, normalization of each sample is required. Specifically, each original ECG sample is first mapped into the range from -1 to 1 by: $Z^E = 2 * (Z^E - \min(Z^E)/(\max(Z^E) - \min(Z^E)) - 1$. They are then changed to have a mean of 0 and a variance of 1 by: $Z^E = (Z^E - \text{mean}(Z^E)/(\text{standard deviation}(Z^E))$.

C. Test

The decompressed ECG signals from the decoder are converted back to their original scale and are then fed into the five pretrained models of [15], [20], [21], [22], and [23] without fine-tuning. For each decompressed ECG sample from the 1st data set, the model of [15] will generate a regression vector and produce the classification results for each of the six abnormality types, with the thresholds of [15] applied. The classification results from both the decompressed ECG signals and the original ECG signals are compared with the actual (i.e., labeled) abnormalities to determine which ones are incorrect. The number of errors (NE) is the difference between the number of incorrect labels obtained with the decompressed signals and the number of incorrect labels using the original ECG signals. A positive value of NE means that the decompressed data predicts a greater number of incorrect labels than the original data. The accuracy (ACC) is the quotient of the number of correct labels from the decompressed data divided by the total number of ECGs. For the 2nd data set, the five major categories used in [22] were tested. The four pretrained models [20], [21], [22], [23] were tested on soft classifier outputs, where no thresholding was applied.

IV. Experiments

A. Training Setting

In the lossy model, there are two types of trainable parameters for each decompression layer, namely, the weights

$(W^E, W^{D0}, W^{D1}, W^F)$ and the scalar parameters γ and β of the affine transformation. The lossy model is trained by the Adam Optimizer [30] with the MSE loss function in PyTorch [31]. In the 1st data set, there are 595 cases used for training, 165 cases for validation, and 67 cases for tests. The same proportion is used for the 2nd data set to obtain 15 722 training cases, 4368 validation cases, and 1747 test cases. The learning rate is 0.001, the batch size is 200, and the number of epochs is 500. As noted earlier, the lossless coding model is nontrainable.

B. Evaluations of the Hybrid Compression

The number of compression/decompression layers in the lossy model depends on its CR. When CR = 2, the number of compression/decompression layers is 1. When CR = 4, the number of layers is 2, and when CR = 8, the number of layers is 3. Twenty separate tests of the data sets are simulated to verify the hybrid compression model. Each test uses different training, validation, and test sets. Since the procedures have some variability, ten instances are simulated for each test, where each instance uses different initial parameter values. The CR, MSE, NE, ACC, and PRD values of one test are the results averaged over these ten instances. Then, the average, best, and worst values of CR, MSE, NE, ACC, and PRD over all 20 tests are computed. The NE and ACC metrics indicate the performance of the compression model in terms of the impact it has on the resulting diagnostic capabilities. The PRD and MSE metrics indicate the differences and errors, respectively, between the original signal and the reconstructed one.

The MSE and PRD for each ECG sample are computed as in (1) and (2), respectively. Here, ORG is the original signal, REC is the decompressed signal, and n is the total number of collected values for one ECG sample (i.e., 4096 samples * 12 leads in this data set). The averaged MSE and PRD of one random test are calculated from all samples in the test set. The MSE and PRD of one permutation test are calculated from ten random tests

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{ORG}(i) - \text{DEC}(i))^2 \tag{1}$$

$$\text{PRD} = 100 \times \sqrt{\frac{\sum_{i=1}^{n}(\text{ORG}(i) - \text{DEC}(i))^2}{\sum_{i=1}^{n}(\text{ORG}(i))^2}}. \tag{2}$$

The calculation of CR for one random test is given in (3). If CR_{lossy} is 4, the total number of bits is 26345472 (i.e., 1024*12*32*67) for the 67 ECG samples of the 1st data set that are available. $\text{CR}_{\text{lossless}}$ is CR from the Huffman coding. The final CR is given by

$$\text{CR} = \text{CR}_{\text{lossy}} * \text{CR}_{\text{lossless}}. \tag{3}$$

The macro AUC error used for the evaluations of the 2nd data set is shown in (4). Here, macro AUC_{ORG} is the macro AUC based on the original signals and macro AUC_{DEC} is based on the decompressed signals. The macro AUC error measures the soft performance of the classifier without considering threshold optimization. The results for the average macro AUC error, as presented in the following section, treat all classes equally since the average of the macro AUC error is taken over all classes

$$\text{macro AUC error} = \frac{\text{macro AUC}_{\text{ORG}} - \text{macro AUC}_{\text{DEC}}}{\text{macro AUC}_{\text{ORG}}} \times 100. \tag{4}$$

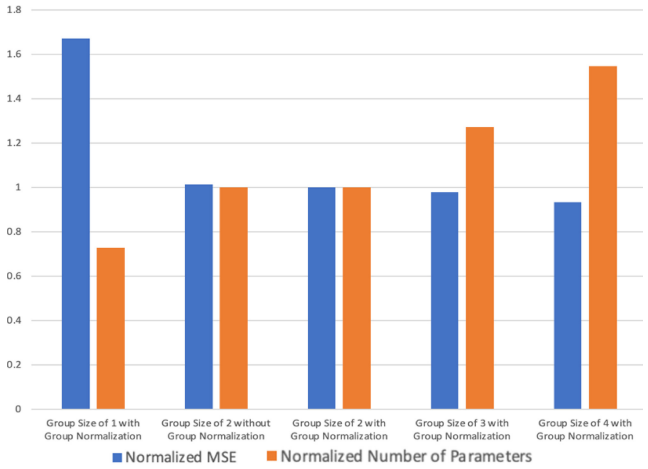

Fig. 4. Comparisons based on the group size. For a group size of 2, results both with and without group normalization in the decoder are shown. Since the group normalization has only a very minor effect on those results, the results for the other group sizes are shown only with group normalization.

V. Results

A. Comparisons of Lossy Models

In the following experiments, ten random tests for one permutation case for the 1st data set were simulated. The number of compression/decompression layers is two in order to obtain a CR of 4 in the lossy model. The averaged and normalized MSEs as well as the number of parameters were determined. Fig. 4 shows the comparison results for different numbers of groups. The first simulations were done for a group size of 2 and, after observing the minimal impact of group normalization on the results in that case, the other group sizes were only evaluated with group normalization. Each group normalization layer only requires the addition of two additional trainable parameters. As the number of groups increases, the MSE very slightly decreases beyond two groups. However, the amount of calculation is significantly increased when there are more than two groups. Therefore, two groups are chosen for the lossy model.

B. Performance of the Hybrid Compression Model for the 1st Data Set

Tables I and II show the performance comparison for three cases corresponding to three CRs in the lossy model when the group number is 2. Specifically, Case 1 is for a lossy model CR of 2, Case 2 is for a lossy model CR of 4, and Case 3 is for a lossy model CR of 8. Here, the final CR, MSE, PRD, NE, and ACC are calculated from the ten averaged random cases among 20 permutation tests. Comparing Cases 1 and 2, the NE is similar but the CR of Case 2 is about twice as high. Comparing Cases 2 and 3, the NE for Case 3 is significantly

TABLE I
COMPARISONS OF CR, MSE, AND PRD FOR THREE COMPRESSION CASES

	CR			MSE			PRD		
	Worst	Average	Best	Worst	Average	Best	Worst	Average	Best
Case 1	2.55	2.60	2.72	0.1670	0.0858	0.0302	28.18	22.93	20.92
Case 2	5.10	5.18	5.41	0.3940	0.2039	0.0771	40.35	37.62	34.66
Case 3	10.02	10.32	10.78	0.5136	0.3186	0.1294	57.48	52.82	50.45

TABLE II
COMPARISONS OF NE AND ACC FOR THREE COMPRESSION CASES

	NE			ACC		
	Worst	Average	Best	Worst	Average	Best
Case 1	2	0.7	-1	0.9876	0.9913	1.00
Case 2	3	0.8	-1	0.9851	0.9910	1.00
Case 3	5	3.4	0	0.9776	0.9846	0.9925

higher. Therefore, Case 2 represents the best overall choice. Although increasing the group size for Case 1 could achieve a better MSE, its average CR is still only 2.6. Also, as shown in Fig. 4, increasing the group size leads to an increase in the number of parameters, which would not be desirable for hardware-limited IoT devices. For case 3, while the group size could be decreased to reduce the computational load, that would also increase its already poor MSE.

We also show a visual comparison between the decompressed ECG signal and the original signal for Case 2. The example shown is the one having the largest MSE among the 200 evaluations (i.e., 20 tests and 10 instances of each test). Figs. 5 and 6 show comparisons at the original scale after calculating the preprocessing parameters. Fig. 5 is the comparison between the original ECG signal and the error. Here, the error is an averaged value of the 67 differences (i.e., for all of the patients) between the decompressed ECG samples and the originals. Fig. 6 is the comparison between the original and decompressed ECG signals. As shown in Fig. 5, the average error is the largest in Leads 3, 5, and 8. This results in the largest difference between the reconstructed signal and the original signal being in those same leads, as shown in Fig. 6. In particular, the reconstructed signals in Leads 3, 5, and 8 of Fig. 6 show fluctuations in the regions outside of the QRS complexes. The fluctuations before the first QRS complex come from the model having learned from other patients' data that have signal information during those times. Future enhancements to the system may be able to minimize these distortions by applying signal pruning or filtering techniques.

Table III shows the comparisons of the computational and memory requirements for three compression cases for both the complete lossy model (i.e., both the encoder and decoder) as well as for the lossy encoder by itself. The estimated total memory size of the complete lossy model for one ECG sample includes the memory required for the inputs, forward/backward pass, and the parameters. The encoder model for Case 2 only has 0.88 million multiply/accumulates (MACs) with 4.12 MB of memory, which is quite suitable for implementation in an embedded system.

By measuring the maximum resident size of the lossless encoder model, we found that it requires about 0.05 MB of memory for each compressed signal from the output of the lossy encoder, which is quite suitable for implementation in an embedded system. Here, this memory estimation is based on the entire lossless encoder, including the calculation of probabilities, the construction of the nodes of the Huffman tree, as well as tree traversing. Compared with the memory consumption of the lossy model as described in the previous paragraph, we conclude that the lossless encoder model requires less memory than the lossy encoder model.

C. Performance of the Hybrid Compression Model for the 2nd Data Set

The Case 2 architecture has also been evaluated using the 2nd data set. Table IV shows the macro AUC percentage error with four DNN classifiers. The proposed compression scheme provides a CR of 4.91 with only a 2.46% averaged macro AUC error. Tables V and VI show the performance of the proposed hybrid classifier on the 2nd data set. The lossy model only has 0.79 million MACs with 3.71 MB of memory required. The compression portion of the system, which is the part that would be implemented in an embedded system, requires only 0.22 million MACs and 1.01 MB of memory for the lossy encoder model, and just 0.03 MB of memory for the lossless encoder model.

D. Comparison Between the Proposed Lossy Model and Other Machine-Learning-Based Compressors

Table VII gives comparisons between the fully connected autoencoder built from [32], the convolutional autoencoder built from [33] and the proposed lossy model for a CR of 4 using the 1st data set. The specific parameters and layers used for the models of [32] and [33] are modified according to our two data sets.

The architecture of the fully connected autoencoder is composed of three models: 1) the flatten model; 2) the encoder model; and 3) the decoder model. The flatten model reshapes the ECG sample data into a long vector (12 leads *4096 samples = 49 152 elements). The input is divided into two equal parts, each having 24 576 samples. Each part passes through the encoder and decoder to form two output tensors. These two output tensors are concatenated and reshaped to obtain 4096 samples on 12 leads. The full details about this model can be found in [32].

The architecture of the convolutional autoencoder is composed of two models: 1) an encoder and 2) a decoder. It considers the 12 lead ECG signals as a gray-scale image. As for the fully connected autoencoder, the ECG inputs are divided into two tensors of 12 leads with 2048 samples. The decoder output concatenates two tensors to obtain

TABLE III
COMPARISONS OF COMPUTATION AND MEMORY FOR THREE COMPRESSION CASES

	Complete Lossy Model (Encoder and Decoder)			Lossy Encoder		
	The Number of Parameters	MACs	Estimated Total Memory Size	The Number of Parameters	MACs	Estimated Total Memory Size
Case 1	2.36 Million	2.36 Million	11.06 Megabytes	0.59 Million	0.59 Million	2.81 Megabytes
Case 2	3.24 Million	3.24 Million	15.19 Megabytes	0.88 Million	0.88 Million	4.12 Megabytes
Case 3	3.69 Million	3.69 Million	17.25 Megabytes	1.03 Million	1.03 Million	4.78 Megabytes

TABLE IV
MACRO AUC ERROR OF FOUR DNNS FOR THE CASE 2 ARCHITECTURE

DNN Classifiers	Macro AUC Error (%)		
	Worst	Average	Best
[20] (Fully Connected Net)	2.78	2.64	2.56
[21] (Inception Net)	3.36	2.72	2.61
[22] (xresnet1d101)	2.39	2.27	2.18
[22, 23] (LSTM)	2.31	2.19	2.06
Average	2.71	2.46	2.35

TABLE V
PERFORMANCE CHARACTERISTICS FOR CASE 2 OF THE HYBRID CLASSIFIER ON THE 2ND DATA SET

Compression Ratio (CR)			Mean Squared Error (MSE)		
Worst	Average	Best	Worst	Average	Best
4.83	4.91	4.98	0.0227	0.0113	0.0077

TABLE VI
ARCHITECTURAL CHARACTERISTICS FOR CASE 2 OF THE HYBRID CLASSIFIER ON THE 2ND DATA SET

Complete Lossy Model (Encoder and Decoder)			Lossy Encoder Model		
Number of Parameters	MACs	Estimated Total Memory Size	The Number of Parameters	MACs	Estimated Total Memory Size
0.79 Million	0.79 Million	3.71 Megabytes	0.22 Million	0.22 Million	1.01 Megabytes

TABLE VII
COMPRESSION MODEL COMPARISONS

	CR	Averaged MSE	Averaged PRD	Averaged NE	Averaged ACC	Number of Parameters	MACs	Total Memory	Averaged MSE Losses (Training/Validation/ Test)
Fully Connected Autoencoder [32]	4	1.3012	96.74	9	0.9701	755.03 Million	1.51 Billion	2880.82 Megabytes	0.0059/0.8710/0.9051
Convolutional Autoencoder [33]	4	1.2907	86.33	9	0.9701	604.00 Million	2.49 Billion	2311.43 Megabytes	0.0095/0.6781/0.6142
Proposed Lossy Method	4	0.0771	34.66	1	0.9900	3.24 Million	3.24 Million	15.19 Megabytes	0.1299/0.1961/0.1481

4096 samples with 12 leads. Each encoder consists of 14 layers and each decoder has 11 layers. The full details about this model can be found in [33].

The factors evaluated include the averaged MSE, averaged PRD, averaged NE, averaged ACC, the number of parameters, the number of MACs, and the total memory required. Ten random instances of one test of the 1st data set were simulated. The results show that the proposed lossy method has the best overall performance considering both the error values and the computational requirements. As shown in the last column of the table, the proposed model exhibits much better generalized data fitting performance in the validation and test phases than the other two models.

VI. CONCLUSION

This article has proposed a novel lightweight ECG compression model for use in a medical IoT system. The proposed hybrid compressor has been tested on two large ECG data sets and the results show that it gives a better tradeoff between CR and accuracy compared with other lossy compression methods. Our design consists of a lossy stage followed by a Huffman encoding stage and it achieves an average CR of 5.18 with a low MSE of 0.20 for the 1st data set and an average CR of 4.91 with a low MSE of 0.01 for the 2nd data set. We have also proposed a novel method for evaluating the errors introduced by the lossy compression that is based on the classification results from DNNs. In other words, we determine if the classifiers are able to reach the same diagnosis on the decompressed data as on the original data so that the distortions introduced by the lossy compression can be considered to be insignificant. Our results show that using the decompressed signals of the 1st data set, the classifier makes an average of only 0.8 additional misdiagnoses out of a total of 402 cases compared to using the original uncompressed signals. In the case of the 2nd

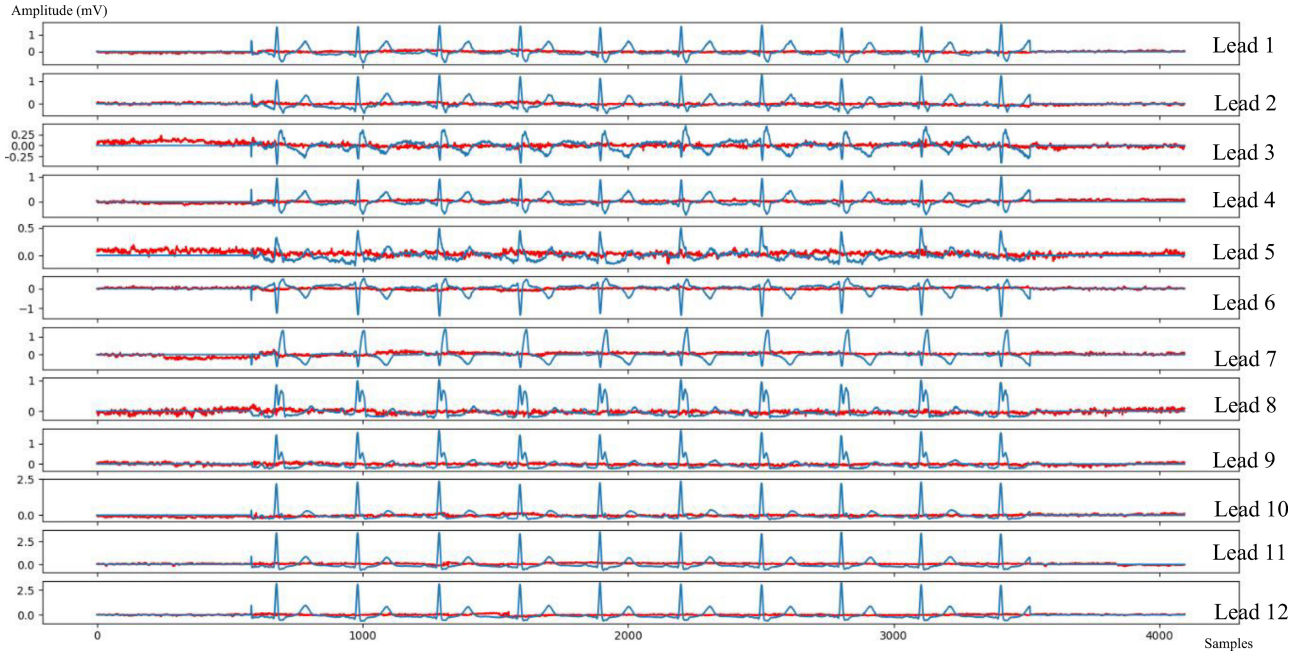

Fig. 5. Comparisons between the original ECG signal (in blue) and the error (in red) at the original scale. Leads 3, 5, and 8 have the largest error magnitudes.

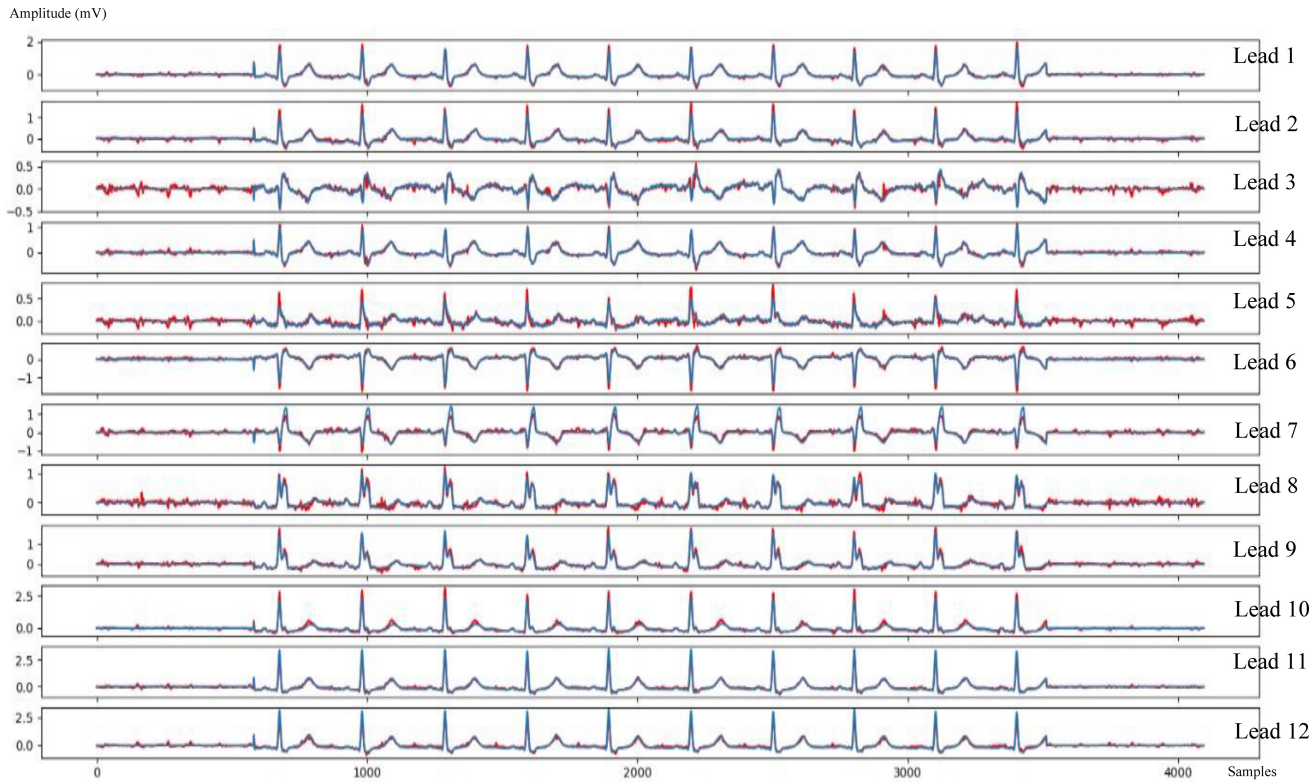

Fig. 6. Comparisons between the original ECG signal (in blue) and the decompressed ECG signal (in red) at the original scale. The distortions in Leads 3, 5, and 8 of the decompressed signal are the most noticeable.

data set, the available comparisons are based on using the macro AUC error criterion. We have found that an average of only 2.46% larger macro AUC error was obtained compared to using the original uncompressed signals. The proposed lightweight lossy compression model has a small memory requirement of 4.12 MB, with 0.88 million parameters, and 0.88 million MAC operations for the 1st data set, and a small memory requirement of 1.01 MB, with 0.22 million parameters, and 0.22 million MAC operations for the 2nd data set. Therefore, the proposed compression system is very suitable for implementation on an embedded platform having limited memory and computation capabilities.

REFERENCES

[1] M. Chan, *Global Status Report on Noncommunicable Diseases 2010*. Geneva, Switzerland: World Health Org., 2011.

[2] P. E. Dilaveris et al., "Simple electrocardiographic markers for the prediction of paroxysmal idiopathic atrial fibrillation," *Amer. Heart J.*, vol. 135, no. 5, pp. 733–738, 1998.

[3] M. Elgendi, A. Mohamed, and R. Ward, "Efficient ECG compression and QRS detection for E-health applications," *Sci. Rep.*, vol. 7, no. 1, p. 459, 2017.

[4] K. C. Wong et al., "User perceptions and experiences of a handheld 12-lead electrocardiographic device in a clinical setting: Usability evaluation," *JMIR Cardio*, vol. 5, no. 2, 2021, Art. no. 21186.

[5] T.-H. Tsai and F.-L. Tsai, "Efficient lossless compression scheme for multi-channel ECG signal," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2019, pp. 1289–1292.

[6] T.-H. Tsai and W.-T. Kuo, "An efficient ECG lossless compression system for embedded platforms with telemedicine applications," *IEEE Access*, vol. 6, pp. 42207–42215, 2018.

[7] K. Bannajak, N. Theera-Umpon, and S. Auephanwiriyakul, "Signal acquisition-independent lossless electrocardiogram compression using adaptive linear prediction" *Int. J. Environ. Res. Public Health*, vol. 20, no. 3, pp. 2753–2768, 2023.

[8] J. Shi et al., "New ECG compression method for portable ECG monitoring system merged with binary convolutional auto-encoder and residual error compensation," *Biosensors*, vol. 12, no. 7, p. 524, 2022.

[9] R. Cervigón, B. McGinley, D. Craven, M. Glavin, and E. Jones, "The effects of compression on the detection of atrial fibrillation in ECG signals," *Appl. Sci.*, vol. 11, no. 13, p. 5908, 2021.

[10] A. Burguera, "Fast QRS detection and ECG compression based on signal structural analysis," *IEEE J. Biomed. Health Inform.*, vol. 23, no. 1, pp. 123–131, Jan. 2019.

[11] M. S. Sundari, S. Loganathan, and C. M. Sujatha, "Hardware implementation of ECG signal compression using SPIHT," in *Proc. J. Phys. Conf. Series*, 2022, Art. no. 12017.

[12] M. L. Talbi and P. Ravier, "Flexible ECG signal modeling and compression using alpha stable functions," *Med. Eng. Phys.*, vol. 109, Nov. 2022, Art. no. 103865.

[13] G. Campobello, A. Segreto, S. Zanafi, and S. Serrano, "An efficient lossless compression algorithm for electrocardiogram signals," in *Proc. Eur. Signal Process. Conf. (EUSIPCO)*, 2018, pp. 777–781.

[14] K. Nemati and K. Ramakrishnan, "Hybrid lossless and lossy compression technique for ECG signals," in *Proc. 3rd Int. Conf. Sens., Signal Process. Secur. (ICSSS)*, 2017, pp. 450–455.

[15] A. H. Ribeiro et al., "Automatic diagnosis of the 12-lead ECG using a deep neural network," *Nat. Commun.*, vol. 11, no. 1, p. 1760, 2020.

[16] P. Wagner et al., "PTB-XL a large publicly available electrocardiography dataset," *Sci. Data*, vol. 7, no. 1, p. 154, 2020.

[17] Y. Chang and G. E. Sobelman, "Lightweight CNN frameworks and their optimization using evolutionary algorithms," in *Proc. Int. Electr. Eng. Congr. (iEECON)*, 2022, pp. 1–4.

[18] A. Vaswani et al., "Attention is all you need," 2017, *arXiv:1706.03762v5*.

[19] K. Weimann and T. O. F. Conrad, "Transfer learning for ECG classification," *Sci. Rep.*, vol. 11, no. 1, p. 5251, 2021.

[20] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 1578–1585.

[21] H. I. Fawaz et al., "InceptionTime: Finding AlexNet for time series classification," *Data Min. Knowl. Disc.*, vol. 34, pp. 1936–1962, Sep. 2020.

[22] N. Strodthoff, P. Wagner, T. Schaeffter, and W. Samek, "Deep learning for ECG analysis: Benchmarks and insights from PTB-XL," *IEEE J. Biomed. Health Inform.*, vol. 25, no. 5, pp. 1519–1528, Sep. 2020.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] Y. Chang and G. E. Sobelman, "Simpler and enhanced multifactorial evolutionary algorithms for continuous optimization tasks," in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IoTaIS)*, 2021, pp. 62–66.

[25] Y. Chang, G. E. Sobelman, and X. Zhou, "Genetic architecture search for binarized neural networks," in *Proc. IEEE Int. Conf. ASIC (ASICON)*, 2019, pp. 1–4.

[26] Y. Chang and G. E. Sobelman, "The class algorithm: Evolution based on division of labor and specialization," in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IoTaIS)*, 2022, pp. 209–215.

[27] Y. Wu and K. He, "Group normalization," 2018, *arXiv:1803.08494v3*.

[28] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[29] D. Lemire and L. Boytsov, "Decoding billions of integers per second through vectorization," *Softw. Pract. Exp.*, vol. 45, no. 1, pp. 1–29, 2013.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.

[31] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," 2019, *arXiv:1912.01703*.

[32] F. Chollet. "The Keras blog." Keras Blog ATOM. Accessed: Dec. 30, 2022. [Online]. Available: https://blog.keras.io/building-autoencoders-in-keras.html

[33] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. Int. Conf. Artif. Neural Netw.*, 2011, pp. 52–59.

Yangyang Chang (Graduate Student Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Minnesota-Twin Cities, Minneapolis, MN, USA, in 2023.

As of the publication of this article, he has published more than ten papers as the first author and serves as the principal engineer of a listed company. His main research directions are lightweight machine learning, VLSI machine learning, neural network architecture search, artificial intelligence application in medical equipment, evolutionary computing, computer vision, and optimization algorithms.

Gerald E. Sobelman (Life Senior Member, IEEE) received the B.S. degree in physics from the University of California at Los Angeles, Los Angeles, CA, USA, in 1974, and the M.S. and Ph.D. degrees in physics from Harvard University, Cambridge, MA, USA, in 1976 and 1979, respectively.

He is a Professor with the Department of Electrical and Computer Engineering, University of Minnesota-Twin Cities, Minneapolis, MN, USA, where he has served as the Director of Graduate Studies for the Graduate Program in Computer Engineering. He has been a Postdoctoral Researcher with The Rockefeller University, New York, NY, USA, and he has held senior engineering positions with Sperry Corporation, Eagan, MN, USA, and Control Data Corporation, Minneapolis. He has authored or coauthored more than 150 technical papers and one book, and he holds 12 U.S. patents. His current research interests are in the areas of circuit and system design for applications in machine learning, communications, and signal processing.

Prof. Sobelman received the Charles E. Bowers Faculty Teaching Award from the College of Science and Engineering of the University of Minnesota. He also received the Graduate and Professional Education Teaching Award from the University of Minnesota. He was the Chair of the Technical Committee on Circuits and Systems for Communications of the IEEE Circuits and Systems Society, and he has also served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS and IEEE SIGNAL PROCESSING LETTERS. In addition, he has chaired many sessions at international conferences in the areas of communications and VLSI design. He has developed and presented short courses on digital VLSI design at several universities and industrial sites, and he has been a consultant to a number of companies. He has served as a Distinguished Lecturer of the IEEE Circuits and Systems Society for three terms. He has been a member of the technical program committees for several IEEE conferences, including ISCAS, SOCC, and ICCSC. He is a member of the Academy of Distinguished Teachers at the University of Minnesota.