Robust and Lightweight Ensemble Extreme Learning Machine Engine Based on Eigenspace Domain for Compressed Learning

Huai-Ting Li, Student Member, IEEE, Ching-Yao Chou, Student Member, IEEE, Yi-Ta Chen, Sheng-Hui Wang, Student Member, IEEE, and An-Yeu Wu, Fellow, IEEE

Abstract—Compressed sensing (CS) is applied to electrocardiography (ECG) telemonitoring system to address the energy constraint of signal acquisition in sensors. In addition, on-sensor-analysis transmitting only abnormal data further reduces the energy consumption. To combine both advantages, “On-CS-sensor-analysis” can be achieved by compressed learning (CL), which analyzes signals directly in compressed domain. Extreme learning machine (ELM) provides an effective solution to achieve the goal of low-complexity CL. However, single ELM model has limited accuracy and is sensitive to the quality of data. Furthermore, hardware non-idealities in CS sensors result in learning performance degradation. In this work, we propose the ensemble of sub-eigenspace-ELM (SE-ELM), including two novel approaches: 1) We develop the eigenspace transformation for compressed noisy data, and further utilize a subspace-based dictionary to remove the interferences, and 2) Hardware-friendly design for ensemble of ELM provides high accuracy while maintaining low complexity. The simulation results on ECG-based atrial fibrillation show the SE-ELM can achieve the highest accuracy with 61.9% savings of the required multiplications compared with conventional methods. Finally, we implement this engine in TSMC 90 nm technology. The postlayout results show the proposed CL engine can provide competitive area- and energy-efficiency compared to existing machine learning engines.

Index Terms—Compressed learning, extreme learning machine, machine learning, noise tolerance, very large scale integration (VLSI).

I. INTRODUCTION

WITH the advances in machine learning techniques and wearable devices, real-time healthcare monitoring can be realized by processing streaming physiological signals obtained from different sensors. One of the most important tasks is the electrocardiography (ECG) telemonitoring system [1], since ECG measuring the electrical activity of the heart has been utilized to diagnose various diseases. In practical situations, this system must be both low-latency and high-accuracy to achieve rapid screening or accurate diagnosis [2]. Furthermore, for the wearable devices, small form factor and low energy consumption are two serious issues.

For the low-power front-end design of telemonitoring system, compressive sensing (CS) is a novel technique that addresses the energy constraint [3]. As shown in Fig. 1(a), CS measures the compressed ECG signals, $\hat{x}$, with fewer measurements than Nyquist rate. Therefore, the CS-based sensor can greatly reduce the sampling rate of high-speed analog-to-digital converter devices, and reduce the transmitting power to achieve a 37.1% extension in node lifetime when compared with conventional compression technique [4]. However, if we want to analyze the original signal, $x$, we need to firstly reconstruct the signal by extremely energy-intensive algorithms [5], which have shown to be impractical for sensor nodes. On the other hand, several recent works have proposed the on-sensor-analysis that only transmits abnormal data after simple analytics. (c) Proposed on-CS-sensor-analysis by robust and lightweight inference in compressed domain without reconstruction.

Fig. 1. Different methods for low-power front-end design: (a) conventional CS with reconstruction for inference. (b) on-sensor-analysis only transmits abnormal data after simple analytics. (c) Proposed on-CS-sensor-analysis by robust and lightweight inference in compressed domain without reconstruction.
To achieve extreme low-energy consumption and real-time analysis, in this paper, we aim at the on-CS-sensor-analysis by a combination of both aforementioned methods. As shown in Fig. 1(c), analyzing signals directly in compressed domain can avoid huge computation cost for reconstruction. Compressed learning (CL) has been recently developed to perform learning and classification in the measurement domain without reconstituting the signals [9], [10]. To make CL feasible on sensor nodes, a lightweight inference engine is indispensable.

To reduce the energy consumption for each classification from microjoule to nanojoule, extreme learning machine (ELM), a single-layer feedforward neural network with extremely low complexity and fast training speed [11], has been popular for fast preliminary screening and implemented in different ways [7], [12], [13]. ELM provides an efficient way to achieve rapid screening with low energy consumption; however, implementing CL directly on sensor node with ELM model will face the following challenges:

1) Learnability degradation caused by hardware non-idealities in CS sensor nodes: The sensors suffer from thermal noise, which can be modeled as white Gaussian noise [15]. In addition, the compressed ECG signals are contaminated by various types of interferences [16]. Baseline wander, which makes the analysis of ECG difficult, is particularly considered as a critical problem. The performance of ELM is very sensitive to the quality of data [17]. Therefore, the noise and interference issues are elevating to dominating concerns, especially in the compressed domain.

2) Unfeasible hardware cost to implement ensemble learning in sensor nodes: Learning directly in the compressed domain with ELM, a very simple machine learning model, suffers from degradation in accuracy. By combining multiple ELM models with AdaBoost algorithm [14], the inference accuracy can be improved and achieve the goal of accurate diagnosis. However, implementing ensemble of ELM with previously presented architecture designs can cause huge area overhead which is unfeasible for sensor nodes.

In this work, we propose a robust and lightweight ensemble of ELM design for ECG-based atrial fibrillation detection, including algorithm and architecture, as shown in Fig. 2. First, we propose the eigenspace-ELM (E-ELM) to improve the learnability of data by utilizing eigenspace transformation. It can reduce the required number of ELM classifiers to achieve lightweight model. Furthermore, we observed that the most significant eigenvectors of interference and cleaned ECG signals are almost orthogonal to each other. By taking this insight into consideration, we propose the sub-eigenspace-ELM (SE-ELM) to make the lightweight learning model robust against interference. More importantly, this eigenspace transformation prevents the inference framework from using high-complexity sparse coding method for data reconstruction. The proposed design can not only achieve CL in noisy situations but also be suitable for VLSI implementation, which make on-CS-sensor-analysis feasible. With the hardware-friendly design for ensemble of ELM inference engine, we achieve high inference accuracy while maintaining low complexity. Our detailed contributions are as follows:

1) Sub-Eigenspace transformation for improving the learnability and removing the interferences directly in the compressed domain: In the off-line training stage, we extract detrended ECG signal from the original signal. Then, we apply the proposed two-stage principal components analysis (PCA) on detrended ECG signal to generate the dictionaries. By combining the CS sensing matrix and dictionaries, in the inference stage, we can project compressed signals onto the sub-eigenspace with the most information preserved, as shown in Fig. 2(a). In the sub-eigenspace, the data has better signal-to-noise ratio (SNR), and needs fewer ELM models to achieve target accuracy. In the white noise plus baseline wander case, the proposed SE-ELM can achieve the same accuracy with more than 61.9% savings of the required multiplications compared with conventional ELM-based methods.

2) Hardware-friendly design for the inference stage of eigenspace transformation and ensemble ELM: As shown in Fig. 2(b), to reduce the required multiplications in ELM inference model, we utilize the linear feedback shift register (LFSR) to generate the random input weights. In addition, a hardware sharing architecture is proposed for the eigenspace transformation and ELM engine to mitigate hardware cost. We implement this CL engine in TSMC 90 nm technology. The core size is 0.1475 mm² at 5 MHz operation frequency.

The rest of this paper is organized as follows. In Section II, we present the background on CS and ELM algorithms, and introduce related works of this specific topic area. Section III presents the proposed E-ELM and SE-ELM frameworks. Section IV shows the numerical experiments and analysis of complexity. The hardware architecture and VLSI implementation results are shown in Section V. Finally, we conclude this paper in Section VI.

II. BACKGROUND AND RELATED WORKS

A. Compressive Sensing (CS) [3]

CS measures signals with fewer measurements than Nyquist rate by utilizing the sparsity of signals. Its encoding procedure can be modeled in the matrix form as [15]

\[ \hat{x} = \Phi(x + n), \]

where \( \hat{x} \in \mathbb{R}^{M \times 1} \) is the compressed measurements, \( \Phi \in \mathbb{R}^{M \times N} \) is the CS sensing matrix, whose entries are independent identically distributed (i.i.d) samples from the uniform distribution or the normal distribution [18], \( x \in \mathbb{R}^{N \times 1} \) is the original signal, and \( n \in \mathbb{R}^{N \times 1} \) could be the additive noise or the interference. This sensing framework is very low-cost and shows great benefit to reducing energy consumption in sensor nodes.

As for the reconstruction of \( x \), the underdetermined problem can be solved by sparse coding method since \( x \) is sparse in some domains. However, the computational complexity involved in sparse coding grows exponentially with the signal...
length $N$. It is impractical to implement the reconstruction in the sensor nodes even if using the most efficient approach.

**B. Compressed Learning (CL) [9]**

From the viewpoint of machine learning, the random projection in (1) can be considered as efficient dimensionality reduction from the data domain, $\mathbb{R}^N$, to the compressed measurement domain, $\mathbb{R}^M$. By utilizing the near isometry property satisfied by the sensing matrix in CS, Calderbank et al. stated that CL, learning directly in the measurement domain without paying the cost of reconstruction, is possible [9]. They provided theoretical bounds guaranteeing the accuracy of linear kernel SVM in the measurement domain is close to that in the data domain.

CL can not only avoid the reconstruction but also reduce the cost of the learning process markedly because the dimensionality is smaller ($M < N$). However, the computational complexity of SVM-based CL is not feasible for resource-limited sensors because most non-linear feature extraction algorithms cannot be performed in compressed domain. CL without feature extraction will present significant challenges to realizing SVM-based CL without feature extraction. Instead, we need other extreme lightweight model whose computational complexity for inference is not directly proportional to the data dimension.

**C. Extreme Learning Machine (ELM) [11]**

ELM first proposed by Huang et al. is an algorithm for training single hidden layer feedforward neural network [11]. Different from the gradient-based algorithms that tuning each parameter iteratively, the input weights and the hidden biases of ELM are chosen arbitrarily and the output weights are solved as a simple linear system. Therefore, ELM has extremely high learning speed and tends to provide great generalization performance.

Given $D$ labeled training data $(\mathbf{d}_i, \mathbf{t}_i)$, where $\mathbf{d}_i \in \mathbb{R}^{1 \times n}$ is an $n$-dimension input feature vector and $\mathbf{t}_i \in \mathbb{R}^{1 \times m}$ is the label vector for $m$-class problem for $i$ from 1 to $D$, target matrix $\mathbf{T} \in \mathbb{R}^{D \times m}$ consisting of all $\mathbf{t}_i$, activation function $g(x)$, and the number of hidden neuron $L$, the details of training algorithm are as following steps:

**Step1:** Randomly assign arbitrary $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{in}]^T$ which is the input weight vector connecting the input neurons and $i$th hidden neuron and bias of the $i$th hidden neuron $b_i$, for $i$ from 1 to $L$.

**Step2:** Calculate the hidden layer output matrix $\mathbf{H} \in \mathbb{R}^{D \times L}$ as

$$\mathbf{H} = \begin{bmatrix}
    g(d_1 \cdot a_{11} + b_1) & \ldots & g(d_1 \cdot a_{1L} + b_L) \\
    \vdots & \ddots & \vdots \\
    g(d_D \cdot a_{L1} + b_1) & \ldots & g(d_D \cdot a_{LL} + b_L)
\end{bmatrix}.$$

(2)

**Step3:** Calculate the output weight matrix $\mathbf{\beta}$ as

$$\mathbf{\beta} = \mathbf{H}^T \mathbf{T},$$

(3)

where $\mathbf{H}^T$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

Next, we investigate the structure of ELM classification engine that we concern in this work. In real applications, data comes in as an input vector. We can also define three steps of ELM classification for each input data.

**Step1:** Compute the input weighting multiplication as

$$\mathbf{z} = [\mathbf{d} \ 1] \cdot \mathbf{A},$$

(4)

where $\mathbf{A}$ is the input weight matrix consisting of all $\mathbf{a}_i$ and $b_i$ that are used in training, and it is defined as

$$\mathbf{A} = \begin{bmatrix}
    a_{11} & a_{12} & \ldots & a_{1L} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{n1} & a_{n2} & \ldots & a_{nL} \\
    b_1 & b_2 & \ldots & b_L
\end{bmatrix}.$$  

(5)

**Step2:** Compute the activation function as

$$\mathbf{h} = g(\mathbf{z}).$$

(6)

**Step3:** Compute the output weighting multiplication as

$$\mathbf{y} = \mathbf{h} \cdot \mathbf{\beta}.$$  

(7)
In these steps, \( \mathbf{d} \) is the input vector, \( \mathbf{z} \) is the input vector of the hidden layer, \( \mathbf{h} \) is the output vector of the hidden layer, and \( \mathbf{y} \) is the output vector. For classification, if \( y_j \) is the element with the largest value in \( \mathbf{y} \), the \( i \)th class is the final result of the classifier.

D. Ensemble ELM With AdaBoost Algorithm

AdaBoost algorithm, one of well-known ensemble methods, combines a series of iteratively and relatively trained classifiers to form a stronger model [14]. To increase the diversity among classifiers, each classifier is trained by same data but each of which has different weights, \( w_j \) for \( j \) from 1 to \( D \), in cost function. The weight of each data will be scaled up or down while every time a trained classifier misclassifies the data or not.

In [19], ELM-based AdaBoost (A-ELM) algorithm is proposed. During each iteration of the boosting algorithm, the least squares problem of solving the output weight \( \beta_i \) in (3) for \( i \)th ELM model could be replaced with the derived weighted least squares problem as

\[
\beta_i = (H_i^T W_i H_i)^{-1} H_i^T W_i T,
\]

where \( W_i \) consists of the instance weights of training data in \( i \)th iteration in the AdaBoost algorithm, and it is a diagonal matrix of dimension \( D \times D \) defined as

\[
W_i = \begin{pmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_D \end{pmatrix}.
\]

Then, the ensemble weight in AdaBoost algorithm, \( \alpha_i \) for \( i \) from 1 to \( C \), where \( C \) is number of classifiers, each classifier is decided by the weighted training accuracy. Higher weighted training accuracy will lead to a larger \( \alpha_i \) that is more significant to the final output. After the training algorithm is done, the final output vector of the ensemble model can be computed as

\[
y_{ensemble} = \sum_{i=1}^{C} \alpha_i y_i.
\]

Equation (10) is the element-wise weighted sum of the output vectors of total \( C \) classifiers.

E. Related Works on Noise-Tolerant Machine Learning

The compressed data is sensitive to noise or interferences [20]. For mitigating noise and interference issues in ECG signals, several signal processing techniques have been proposed on original data domain [21], [22]. However, before applying these techniques, we have to firstly reconstruct the compressed data to original data, which should be avoided in the scenario of on-CS-sensor-analysis. Instead, to deal with noise and interference issues in compressed ECG signals without reconstruction, some general machine learning methods can be applied, such as regularized learning models [17] and data-driven hardware resilience (DDHR) [23].

1) Probabilistic Regularized ELM (PR-ELM) [17]: In [17], PR-ELM is proposed to improve model performance when subject to noisy data and outliers. Regularization is a popular process of introducing additional information to solve an ill-posed problem or to prevent overfitting. By adding a penalty on different model parameters, regularization can reduce the model complexity. Hence, the model is less likely to fit the noise in training data. To minimize not only the modeling error variance but also the modeling error mean, Xinjiang et al. proposed a new objective function for training ELM model as

\[
\text{Obj} = \min_{\beta, \varepsilon, \zeta} \frac{1}{2} \| \beta \|^2 + \frac{\gamma}{D^2} \sum_{i=1}^{D} (e_i - \zeta)^2 + C \frac{1}{2} \zeta^2
\]

s.t. \( h_i \cdot \beta = t_i - e_i, \quad i = 1, \ldots, D, \quad (11) \]

where \( e_i \) is modeling error, \( \zeta \) is the global error factor which builds a bridge among errors, and both \( \gamma \) and \( C \) are the regularization parameters that can be decided by cross-validation. This method incorporated the distribution of modeling error into training process, thus allowing for more robust performance than traditional ELM model.

2) Data-Driven Hardware Resilience (DDHR) [23]: To obtain an inference model for sensed signals in the presence of errors caused by hardware non-idealities, Zhuo et al. proposed DDHR approach, which utilizes data-driven training to construct an error-aware model [23]. With DDHR, the effects of interferences can be regarded as altering the statistics of sensed signals in the form of new class distributions. As a result, the inference performance can be retained as long as the mutual information is retained in the error-affected data and the new distributions can be learned by the error-aware model. That is, to make DDHR effective, the data statistics of the training data should match those of testing data.

F. Design Issues in Prior Works

Applying ELM model to perform CL can achieve the goal of extreme low-energy consumption for on-CS-sensor-analysis. However, classifying compressed measurement with ELM, a very simple machine learning model, suffers from degradation in accuracy. Furthermore, the performance of ELM model is very sensitive to the quality of data [17]. Therefore, the noise and interference issues are elevating to dominating concerns, especially in the compressed domain. DDHR approach cannot deal with white Gaussian noise because white noise is a sequence of random numbers that cannot be learned by the error-aware model [23]. PR-ELM only mitigates the effects of non-idealities by modifying the regularization terms in optimization for a given input space [17]. The aforementioned related works only restore the performance to a certain level, which is not enough for an accurate diagnosis, because these methods are lacking in finding or transforming to a more informative and cleaner space for learning tasks. The comparison among related works is shown in Table I.

III. PROPOSED ROBUST ENSEMBLE OF ELM WITH EIGENSPACE TRANSFORMATION

From the viewpoint of feature space transformation, we propose an eigenspace transformation which can be directly...
performed on compressed domain to extract information from noisy data. Furthermore, to remove the interference from compressed signals, we introduce a subspace-based dictionary to project compressed signals onto the sub-eigenspace with the most information preserved. Although the idea of transformation using subspace dictionaries is already proposed in our earlier work [10], in this work, we propose a more systematic approach for obtaining both subspace-based dictionaries and transformation matrix. With the proposed sub-eigenspace transformation, the number of ELM models required in the ensemble method can be reduced. Therefore, compared to the prior works, the proposed SE-ELM can achieve the highest accuracy with lowest computational cost.

A. Proposed Eigenspace-ELM (E-ELM) Framework

To enhance the learning performance by information extraction and noise mitigation, the E-ELM framework is proposed as shown in Fig. 3. It consists of two stages: 1) off-line training and 2) on-line inference. The first stage utilizes a small amount of reconstructed data to learn a dictionary by PCA. Then, we can obtain an eigenspace transformation matrix, which is used in the inference stage, by using this dictionary and sensing matrix of CS. After transforming all training data with the learned transformation matrix, we can train the ELM models in eigenspace. The second stage is directly performed in the measurement domain. The compressed signals are transformed onto eigenspace by the transformation matrix obtained in the training stage. Then, the projected data, s, is further analyzed by the ELM models directly in the eigenspace.

1) PCA-Based Dictionary: The computational complexity of most machine learning inference algorithms is highly correlated to the data dimension. Therefore, dimension reduction is an important stage in on-device inference. Many works [24], [25] have utilized PCA in different applications. PCA utilizes the property that the intrinsic dimension of a dataset is much smaller than the data dimension. The dataset can thus be represented by a small number of vectors, the principle components (PC). PCs are linear transformations of the original set of variables which are orthogonal and ordered by the variation of data on the corresponding basis. By choosing the first few PCs as basis, we can transform the dataset onto a lower dimensional subspace while retaining the most information.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Computational Complexity</th>
<th>Non-idealities Considered</th>
<th>Transformation in Compressed Domain</th>
<th>Accuracy of CL with Non-idealities</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDHR-SVM [23]</td>
<td>High</td>
<td>Yes</td>
<td>No</td>
<td>High</td>
</tr>
<tr>
<td>ELM [11]</td>
<td>Low</td>
<td>No</td>
<td>No</td>
<td>Low</td>
</tr>
<tr>
<td>PR-ELM [17]</td>
<td>Low</td>
<td>Yes</td>
<td>No</td>
<td>Middle</td>
</tr>
<tr>
<td>Proposed SE-ELM</td>
<td>Lowest</td>
<td>Yes</td>
<td>Yes</td>
<td>Highest</td>
</tr>
</tbody>
</table>

While PCA has been used primarily as a sensing matrix [26], in this work, we exploit it as a transformation dictionary instead. Assume a dataset of D reconstructed training data \( \hat{x} = [x_1 x_2 \ldots x_D] \), where each \( x_i \in \mathbb{R}^{N \times 1} \). The first step of PCA is to compute the covariance matrix, \( \sum \), of the dataset. Then, the eigendecomposition of \( \sum \) is performed, and we can obtain a matrix \( U \) whose \( i \)th column is the \( i \)th eigenvector of \( \sum \) with \( i \)th largest eigenvalue. By choosing the first \( k_s \) important eigenvectors, the main data information can be kept and the noise can be mitigated in the low dimensional subspace. The proposed PCA-based dictionary \( \psi \) consists of the first \( k_s \) columns of \( U \) as follows:

\[
\psi = U(:, 1:k_s).
\]

The PCA-based dictionary keeps the most significant information with the smallest dimension. The rest of eigenvectors that are not in \( \psi \) are considered as noise. We utilize the data statistics information to eliminate the redundant basis and build an overdetermined dictionary.

2) Eigenspace Transformation Matrix: To improve the performance of CL without the huge computational overhead of reconstruction, we propose to use an eigenspace transformation matrix, combing the information in PCA-based dictionary and the CS sensing matrix, to transform the compressed signal. For the dataset whose intrinsic dimension is much smaller than the data dimension, the variance of most PCs is close to zero. This means the dataset is sparse in the eigenspace. Therefore, we can find a low dimensional subspace with fixed basis in the sparse space and preserve the most information at the same time. By utilizing the proposed PCA-based dictionary in (12), we can represent the compressed signal, \( \hat{x} \), as

\[
\hat{x} \cong \Phi \psi s = \Theta s.
\]

where \( \Theta \in \mathbb{R}^{M \times k_s} \) is the matrix multiplication of sensing matrix and the proposed PCA-based dictionary, and \( s \) is the representation of \( \hat{x} \) in the eigenspace. To estimate \( s \), we can solve the optimization problem as

\[
\min_s \| \hat{x} - \Theta s \|^2.
\]
Therefore, we can obtain \( \tilde{s} \), the estimated result of \( s \), with the least-square sense as

\[
\tilde{s} = \Theta^T \hat{x} = (\Theta^T \Theta)^{-1} \Theta^T \hat{x}.
\]  

(15)

Consequently, we define the eigenspace transformation matrix \( T \) as

\[
T = \Theta^\dagger = (\Theta^T \Theta)^{-1} \Theta^T.
\]  

(16)

This eigenspace transformation requires only matrix multiplication, because \( \Theta \) in (13) is an overdetermined dictionary which can be solved by least squares method. This matrix is calculated in the off-line training stage, and we can utilize it to perform eigenspace transformation on the received compressed signals in the on-line inference stage.

**B. Proposed Sub-Eigenspace-ELM (SE-ELM) Framework**

The proposed E-ELM can improve the performance of CL under white noise issue because of the benefits from the noise mitigation by PCA. However, when facing the interference issue in compressed domain, the proposed E-ELM degrades in accuracy because the interference spreads onto basis vectors in the PCA-based dictionary. Therefore, as shown in Fig. 4, we propose the SE-ELM to divide the aforementioned PCA-based dictionary into two subspaces for signal information and interference, respectively. In this paper, we choose baseline wander as a case study of interference issue. When dealing with other types of interference, this framework still works well as long as we replace the baseline extraction block in Fig. 4 with proper preprocessing methods.

1) Two-Stage PCA-Based Dictionary: In the off-line training stage, we utilize the reconstructed data contaminated by interference to learn the dictionaries. First, we find the PCA-based dictionary, \( \psi_E \), as discussed in Sec. III.A by using the contaminated data in the original domain. Therefore, some vectors in the basis of the dictionary contain the statistical information of interference.

On the other hand, the contaminated data in original domain can be handled by conventional interference removal methods. In this work, we exploit wavelet adaptive filtering (WAF) [22] to decompose the ECG signal and baseline wander. The ECG signal of training set is decomposed up to 7 levels using Wavelet Transform (WT). The frequency of the 7th level approximation coefficients is in the range of 0-1.4 Hz, in which consists of baseline signals. Thus, the 7th level approximation coefficients are optimized by an adaptive filter. The filtered output and the detail coefficients are used for inverse WT. Then, we can obtain the detrended data.

Next, we apply the transpose of the PCA-based dictionary, \( \psi_E^T \), to project all the \( D \) detrended data in the training set \( X_d \in \mathbb{R}^{N \times D} \) to \( Y_d \in \mathbb{R}^{k_s \times D} \) as

\[
Y_d = \psi_E^T X_d.
\]  

(17)

To divide this projected eigenspace into two subspaces, signal space and baseline space, we apply the PCA again on \( Y_d \). As discussed in Sec. III.A.1), we can obtain a matrix \( U_Y \in \mathbb{R}^{k_s \times k_s} \) whose \( i \)th column is the \( i \)th eigenvector of the covariance matrix of \( Y_d \). Because \( Y_d \) is projected from the detrended data, the more important eigenvectors are regarded as the basis of projected signal space. On the other hand, the rest of eigenvectors are regarded as the basis of projected baseline space. That is, we can separate the matrix \( U_Y \) into two sub-matrices for representing signal and baseline as

\[
U_s = U_Y(:, : \cdot \cdot \cdot : k_s)
\]  

and

\[
U_b = U_Y(:, : k_s + 1 : k_s),
\]  

(19)

where \( k_s \) is the dimension of signal space, decided by the percentage of cumulative variance. Then, as shown in Fig. 5, the signal in original domain can be represented as

\[
x \cong \psi_E [U_s \ U_b] \begin{bmatrix} s_s \\ s_b \end{bmatrix} = [\psi_s \ \psi_b] \begin{bmatrix} s_s \\ s_b \end{bmatrix},
\]  

(20)

where \( s_s \) is the signal component of the contaminated data in the eigenspace, \( s_b \) is the baseline component of the contaminated data in the eigenspace, and both \( \psi_s = \psi_E U_s \) and \( \psi_b = \psi_E U_b \) are the proposed subspace-based dictionaries for signal and baseline wander, respectively.

2) Sub-Eigenspace Transformation Matrix: By utilizing the proposed subspace-based dictionaries, we can represent the compressed signal, \( \hat{x} \), as

\[
\hat{x} \cong \Phi [\psi_s \ \psi_b] \begin{bmatrix} s_s \\ s_b \end{bmatrix} = [\Theta_s \ \Theta_b] \begin{bmatrix} s_s \\ s_b \end{bmatrix},
\]  

(21)

where \( \Theta_s = \Phi \psi_s \in \mathbb{R}^{M \times k_s} \) and \( \Theta_b = \Phi \psi_b \in \mathbb{R}^{M \times (k_s - k_s)} \). We can solve both \( s_s \) and \( s_b \) with the least-square sense described in (13), (14), and (15). However, for on-line inference stage, we implement the machine learning on only \( s_s \). Transforming \( \hat{x} \) to \( s_s \) can be viewed as baseline removal on compressed domain because the baseline component is left in \( s_b \). According to [27], the Moore-Penrose inverse of
a columnwise partitioned matrix can be particularly formulated as
\[
[\Theta_s \Theta_b]^\top = [(P_{\Theta_b} \Theta_s)\top (P_{\Theta_b} \Theta_b)]
\]
(22)
where the orthogonal projectors are specified as
\[
P_{\Theta_s} = I_M - \Theta_s \Theta_s^\top
\]
(23)
and
\[
P_{\Theta_b} = I_M - \Theta_b \Theta_b^\top
\]
(24)

Therefore, we can derive \(\tilde{s}_n\), the estimated results of \(s_n\), as
\[
\tilde{s}_n = (P_{\Theta_b} \Theta_s)^\top \hat{x} = T_{SE} \hat{x}
\]
(25)
where \(T_{SE} = (P_{\Theta_b} \Theta_s)^\top \in \mathbb{R}^{k_s \times M}\) is the proposed sub-eigenspace transformation matrix. With this matrix, we can transform the compressed measurement \(\hat{x}\) to the signal component \(\tilde{s}_n\) without the computational cost for calculating the baseline component, which reduces the required multiplications in transformation by \((k_s - k_{ss}) \times M\).

The proposed SE-ELM framework benefits from the conventional interference removal methods in the original data domain without the huge overhead of reconstruction in the on-line inference stage. With the proposed sub-eigenspace transformation matrix, we can achieve the effect of interference removal directly on the compressed domain.

IV. NUMERICAL EXPERIMENTS AND ANALYSIS OF COMPUTATIONAL COMPLEXITY

A. Experimental Settings for Different Noisy Scenarios

ECG-based atrial fibrillation (AF) detection is used to validate our proposed frameworks for sensor-data binary classification. The raw ECG signals we used were recorded from the intensive care unit (ICU) of stroke in National Taiwan University Hospital (NTUH) with sampling frequency of 512 Hz. We set 1 second as input dimension; therefore, each data is a time series of 512 samples containing baseline wander. The data were visually checked and labeled as AF or non-AF by doctors. For each label, there are 2500 training data and 1000 inference data. To obtain the compressively-sensed signal by doctors. For each label, there are 2500 training data and 1000 inference data. The raw ECG signals we used were recorded from the intensive care unit (ICU) of stroke in National Taiwan University Hospital (NTUH) with sampling frequency of 512 Hz.

There are three different noisy scenarios in the experiments, as shown in Table IV, including the ideal case, the white noise case, and the worst case, white noise plus baseline wander case. In the ideal case, the recorded raw ECG signals are processed by the interference removal algorithms. Therefore, the input signal in the ideal case is the detrended ECG signals, \(x_d\). In the white noise case, to simulate the thermal noise in CS sensors, we add additional white Gaussian noise under different SNR to the detrended ECG signals. In the worst case, we add additional white Gaussian noise under different SNR to the raw ECG signals, which is equal to adding both interference and white Gaussian noise to the detrended ECG signals.

For the analysis of the computational complexity, we calculated the required number of multiplication in the inference stage, as discussed in Section IV.B.

We analyze the computational complexity of SVM, PR-ELM, and the proposed frameworks for inference on the compressed domain. Because the input vector to each model is the compressed signal, the input dimension is \(M\). For SVM with RBF kernel, the kernel size is decided by the number of support vectors. Therefore, the computational complexity for RBF-SVM provided by [8] is \(\#SV \times M\), where \(\#SV\) is the number of support vectors.

### Table II
**EXPERIMENTAL SETTINGS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>NTUH ICU ECG data</td>
</tr>
<tr>
<td>Sampling Frequency</td>
<td>512 Hz</td>
</tr>
<tr>
<td>Input Dimension (N)</td>
<td>512 (1 sec. ECG)</td>
</tr>
<tr>
<td>Number of Classes</td>
<td>2 (AF/non-AF)</td>
</tr>
<tr>
<td>Number of training data for each class</td>
<td>2500</td>
</tr>
<tr>
<td>Number of inference data for each class</td>
<td>1000</td>
</tr>
</tbody>
</table>

### Table III
**PARAMETERS SETTINGS FOR MACHINE LEARNING MODELS**

<table>
<thead>
<tr>
<th>SVM (LIBSVM [28])</th>
<th>PR-ELM [17]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel</strong></td>
<td><strong>Gamma ((\gamma))</strong></td>
</tr>
<tr>
<td>Radial Basis Function (RBF)</td>
<td>1, 5, 8</td>
</tr>
<tr>
<td><strong>Cost ((C))</strong></td>
<td>10^2 to 10^4\times 2500</td>
</tr>
</tbody>
</table>

### Table IV
**DIFFERENT NOISY SCENARIOS**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Measurement from CS sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal Case</td>
<td>(\hat{x} = \Phi(x_d))</td>
</tr>
<tr>
<td>White Noise Case</td>
<td>(\hat{x} = \Phi(x_d + n_{white}))</td>
</tr>
<tr>
<td>Worst Case (White Noise + Baseline Wander)</td>
<td>(\hat{x} = \Phi(x_d + n_{white} + n_{baseline}))</td>
</tr>
</tbody>
</table>
As for the ELM-based models, the required multiplications are $n \times L$ for ELM input weighting multiplications, and $L \times m$ for ELM output weighting multiplications, where $n$ is the input dimension of ELM model, $L$ is the number of hidden neurons, and $m$ is the number of classes. With the proposed hardware-friendly ELM design detailed in Section V.A, we can complete the input weighting multiplications with only adders and multiplexers. Therefore, we can ignore the $n \times L$ term standing for input weighting multiplications. Furthermore, for binary classification problem, because one column in the output weight matrix $\beta$ in (3) is the additive inverse of the other column, the computational complexity can be further reduced to only $L$. Consequently, for all the $C$ classifiers in the ensemble ELM models, the number of required multiplications is $L \times C$.

In addition, the overheads of the proposed E-ELM and SE-ELM in inference stage are the matrix multiplications for eigenspace transformation in (15) and sub-eigenspace transformation in (25). Therefore, the number of required multiplications of E-ELM and SE-ELM are $M \times k_s + L \times C_E$ and $M \times k_{ss} + L \times C_{SE}$, respectively. $k_s$ and $k_{ss}$ are the dimensions of eigenspace and sub-eigenspace. $C_E$ and $C_{SE}$ are the required number of ELM models for E-ELM and SE-ELM.

### C. Evaluation of E-ELM Framework

First, we evaluated the enhancement of learning performance provided by the E-ELM framework in the ideal case, as shown in Fig. 6. As a powerful and high-complexity model, one SVM model can achieve an accuracy of 94.56%. However, the required number of multiplications in SVM model is extremely large, as shown in Table V. The dotted lines with different colors in Fig. 6 show the inference accuracy over the increasing number of ELM classifiers achieved by AdaBoost-ELM with different $L$. The inference accuracy increases as we combine more models in the AdaBoost-ELM. The solid lines in Fig. 6 show the inference accuracy achieved by the proposed ensemble of E-ELM, which is superior to conventional AdaBoost-ELM. This improvement benefits from the signal information extraction by PCA. In the proposed E-ELM, the dimension of PCA-based dictionary, $k_s$, was chosen as 39 with the criterion that the percentage of cumulative variance is larger than 85%. When $L$ is equal to 400, the proposed E-ELM with 50 classifiers can achieve the inference accuracy of 93.51%, which is a competitive performance compared to SVM. As shown in Table V, E-ELM obtains 94.07% saving of the required multiplications compared to SVM model in the ideal case.

Fig. 7 presents the comparison of the inference accuracy over the increasing number of ELM classifiers between different frameworks in the white noise case under different SNR. Although we only show the case where $L$ is equal to 400, the simulations with different settings of $L$ show a consistent performance. PR-ELM considers both modeling error variance and modeling error mean in the training stage. Therefore, compared to AdaBoost-ELM, AdaBoost-PR-ELM has higher inference accuracy when the compressed signals are contaminated by Gaussian white noise. The proposed E-ELM outperforms the PR-ELM, because the eigenspace transformation can achieve signal information extraction and noise mitigation at the same time. As shown in Table VI, the SNR measured in the eigenspace is much higher than that in the compressed measurement domain when we inject the same level of white Gaussian noise, which illustrates that the learnability of the compressed noisy data is improved by the proposed eigenspace transformation.

The proposed E-ELM framework can show competitive inference accuracy as discussed above, which is accurate enough for fast screening. More importantly, the required multiplications by both the ensemble of ELM-based models, E-ELM and PR-ELM, are significantly less than that by SVM. Although E-ELM needs the overhead for eigenspace transformation, it requires much fewer classifiers in the ensemble method to achieve the same inference accuracy compared to PR-ELM. Consequently, the overall required multiplication ($M \times k_s + L \times C_E$) is reduced. To achieve the inference accuracy provided by 50 PR-ELM classifiers, as shown in Table VII, E-ELM only needs 23, 19, 16, and 9 classifiers when SNR is 20, 15, 10, and 5 dB, respectively. The results show that E-ELM is more robust against the white noise when the SNR is worse. Therefore, compared to the PR-ELM, the proposed E-ELM obtains 41.5% saving of the required multiplications averagely in different SNR levels.

### D. Evaluation of SE-ELM Framework

Fig. 8 shows the comparison of the inference accuracy over the increasing number of ELM classifiers between
Fig. 7. Comparison of the inference accuracy over the increasing number of ELM classifiers between SVM, E-ELM, PR-ELM, and ELM models with 400 hidden neurons in the white noise case when (a) SNR is 5, (b) SNR is 10, (c) SNR is 15, and (d) SNR is 20.

<table>
<thead>
<tr>
<th>Injected SNR (dB)</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required classifiers (C_E)</td>
<td>9</td>
<td>16</td>
<td>19</td>
<td>23</td>
</tr>
<tr>
<td>Required multiplications by PR-ELM (L × C)</td>
<td>20000</td>
<td>20000</td>
<td>20000</td>
<td>20000</td>
</tr>
<tr>
<td>Required multiplications by E-ELM (M × k_s + L × C_E)</td>
<td>8592</td>
<td>11392</td>
<td>12592</td>
<td>14192</td>
</tr>
<tr>
<td>Saving of multiplications</td>
<td>57%</td>
<td>43%</td>
<td>37%</td>
<td>29%</td>
</tr>
</tbody>
</table>

**TABLE VIII**

The Required Number of Classifiers in ELM-Based Models to Achieve the Accuracy Provided by DDHR-SVM and the Comparison of the Required Multiplications in the Worst Case

<table>
<thead>
<tr>
<th>Injected SNR (dB)</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required classifiers in PR-ELM (C_{PR})</td>
<td>&gt;50</td>
<td>&gt;50</td>
<td>&gt;50</td>
<td>&gt;50</td>
</tr>
<tr>
<td>Required classifiers in E-ELM (C_E)</td>
<td>35</td>
<td>45</td>
<td>&gt;50</td>
<td>&gt;50</td>
</tr>
<tr>
<td>Required classifiers in SE-ELM (C_{SE})</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Required multiplications by PR-ELM (L × C_{PR})</td>
<td>&gt;20000</td>
<td>&gt;20000</td>
<td>&gt;20000</td>
<td>&gt;20000</td>
</tr>
<tr>
<td>Required multiplications by E-ELM (M × k_s + L × C_E)</td>
<td>18992</td>
<td>22992</td>
<td>&gt;24992</td>
<td>&gt;24992</td>
</tr>
<tr>
<td>Required multiplications by SE-ELM (M × k_{ss} + L × C_{SE})</td>
<td>6512</td>
<td>7712</td>
<td>8112</td>
<td>8112</td>
</tr>
</tbody>
</table>

Different frameworks in the worst case under different SNR. By comparing Fig. 7 and Fig. 8, we can observe that SVM, E-ELM, PR-ELM, and ELM suffer certain accuracy degradation because of the interference in the compressed signals. To improve the performance of SVM model, we applied the DDHR concept to it. Even though the interference effect on signals can be learned to a certain degree by the error-aware model, DDHR-SVM has only about 2% higher accuracy than SVM. This is because the space for learning task still contains the learning task-unrelated information about interference. This reason also explains the accuracy degradation in PR-ELM and ELM. On the other hand, in E-ELM, even though the compressed signals are transformed to the eigenspace, the interference component spreads out on the eigenspace learned from the cleaned data. Therefore, the proposed eigenspace division in the SE-ELM is required when there are some unnecessary components with specific distribution in the compressed signals.

The SE-ELM framework removes the interference directly in the compressed domain, and thus it can achieve the highest inference accuracy in the worst case. There is a tradeoff between the achieved accuracy and energy savings. As shown in Fig. 7 and Fig. 8, the ensemble methods can achieve higher accuracy with more used models. In the proposed SE-ELM, the dimension of signal space, k_{ss}, was chosen as 29 with the criterion that the percentage of cumulative variance is larger than 85%. As shown in Fig. 8, when L is equal to 400, SE-ELM needs only 7, 10, 11, and 11 classifiers to achieve the inference accuracy provided by DDHR-SVM when SNR is 20, 15, 10, and 5 dB, respectively. With the parameter values, we can obtain and summarize the required multiplications in Table VIII. For example, the required number of multiplications by SE-ELM is 128 × 29 + 400 × 11 = 8112 when SNR is 20. Therefore, SE-ELM obtains more than 66.8% and 61.9% savings of the required multiplications averagely in different SNR levels compared to E-ELM and PR-ELM, respectively, and can achieve the goal of lowest computational complexity for on-CS-sensor-analysis.

V. ARCHITECTURE DESIGN AND VLSI IMPLEMENTATION

In this work, the main concern of the hardware architecture design is an area- and energy-efficient implementation for the inference stage of the proposed SE-ELM framework. To reduce the hardware cost for implementing A-ELM, we propose a hardware-friendly ensemble of ELM. The implementation results show that the proposed engine is not only reconfigurable for the number of classifiers but also cost-effective.

A. Hardware-Friendly Ensemble of ELM

We adopt the cost-sensitive method in [19] to achieve A-ELM. To further increase the diversity among different ELM classifiers, in each iteration of the AdaBoost algorithm, we assign different input weight matrix A_i to i-th ELM model, as shown in Fig. 9. With different input weight matrices, the hidden layer output matrices H’s in different iteration have different values even though the training data is the same. Therefore, the obtained output weight matrices B’s for different ELM models are more diverse and the ensemble model is more accurate. However, it needs to compute the different input weighting multiplications in (4), which results in the additional k_{ss} × L × C multiplications. The input
weighting multiplications account for the highest proportion in the overall required multiplications.

To make the input weighting multiplication more hardware-friendly, we construct the input weight matrices as random Bernoulli matrices, where each column is either 1 or -1. By doing so, we can use only adders and multiplexers to complete the input weighting multiplication without significant drop in accuracy. Furthermore, if we store all the random Bernoulli matrices in the inference engine, it requires additional memory buffers. During the eigenspace transformation stage, the multipliers reads the input data from the input shift registers and the registers, in the eigenspace transformation stage, the multiplier computes the element-by-element multiplication. In addition, these two computations are performed at different execution stages. Therefore, to reduce the number of implemented multipliers, we design a hardware reuse method with folding technique. As shown in Fig. 10(a), the multiplexers, controlled by the control unit (DCU) (Fig. 10(d)). In addition, Fig. 11 shows the execution flow of the proposed SE-ELM engine. It consists of five main execution stages: configuring stage, data collecting stage, eigenspace transformation, ELM computing stage, and weighted voting for final results.

1) Hardware Sharing Architecture for Eigenspace Transformation and ELM Output Weighting Multiplication: The eigenspace transformation and ELM output weighting multiplication are both matrix multiplication. In addition, these two computations are performed at different execution stages. Therefore, to reduce the number of implemented multipliers, we design a hardware reuse method with folding technique. As shown in Fig. 10(a), the multipliers, controlled by the DCU, at the inputs of the multiplier select proper inputs to compute the element-by-element multiplication.

After 128 cycles for collecting the input data into 128 shift registers, in the eigenspace transformation stage, the multiplier reads the input data from the input shift registers and the $T_{SE}$ memory buffers. During the eigenspace transformation stage, the compressed data cyclically shifts into the 128 registers. On the other hand, the multiplier reads the input data from the output of the sigmoid unit and the $\beta$ memory buffers in the ELM computing stage.

2) ELM Input Weighting Multiplication (EIW): EIW computes the input weighting multiplication and the activation function in ELM algorithm, as described in (4) - (6). Since the shared multiplier can complete one of the element-by-element multiplications in the output weighting multiplication, EIW block must generate the result of one hidden neuron in each
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

LI et al.: ROBUST AND LIGHTWEIGHT ENSEMBLE ELM ENGINE BASED ON EIGENSPACE DOMAIN FOR CL

Fig. 10. Proposed hardware architecture design for the ensemble of SE-ELM, including (a) hardware sharing for eigenspace transformation and ELM output weighting multiplication, (b) ELM input weighting multiplication (EIW), (c) ensemble output unit, and (d) data control unit (DCU).

Clock cycle. Therefore, we utilize the multiplexers and adder tree without hardware sharing. On the other hand, because the clock frequency is fast enough for real-time ECG-based AF detection, we do not insert any pipeline registers in the adder tree to save hardware costs. Although transformation requires additional \( k_{ss} \) registers used for storing the transformed data, it greatly reduces the size of adder tree in the ELM input weighting multiplication. Therefore, from the whole framework perspective, the eigenspace transformation reduces the hardware complexity.

As shown in Fig. 10(b), for the input weighting multiplications, to reduce the computational complexity and memory requirements, we utilize the LFSR to generate the input weight matrix. Since \( k_{ss} \) is chosen as 29 in this design, we implemented a 30-bit LFSR where the values in the 30 registers are used as one of the column vectors in the input weight matrix in (5). In each clock cycle, these 30 binary values control the multiplexers in EIW to perform multiplication with 1 or \(-1\). Therefore, we can complete the multiplications by multiplexers and adders. As for the activation function, we implemented the sigmoid unit using the piecewise linear approximation [29].

3) Ensemble Output Unit: After collecting the results from different ELM models, the ensemble output unit will compute the ensemble average in (10) to obtain the final output. For binary classification, the classification result of each ELM model is the signed bit of the result of output weighting multiplication. The DCU will provide the appropriate ensemble weight \( \alpha \) for corresponding ELM model. Then, the ensemble average can be obtained from the accumulation of the ensemble weights selected by the most significant bit (MSB) of each output weighting multiplication result, and finally we can obtain the final classification result from the MSB of the accumulated ensemble average.

4) Data Control Unit (DCU): The control mechanism in this design is based on counters. With the configuration of model parameters, DCU controls the multiplexers in different parts of this architecture to decide the data flow, including 128 compressed input data in Fig. 10(a), 29 transformed data in Fig.10(b), and the resets of the accumulations in both Fig.10(a) and Fig.10(c). DCU also controls the memory addresses to select the model parameters for specified ELM or eigenspace transformation. To achieve the reconfigurable design, DCU also decides the termination of ensemble method according to the specified number of classifiers, and then lets the input buffer collect new data.

C. Execution Flow of the Proposed Engine

The execution flow of the proposed engine is shown in Fig. 11. First, in the configuring stage, the parameters...
components are kept in the memory buffers. After each 128 cycles, one of the transformed TSE eigenspace transformation is completed by the multiplier with a proper model parameter fed from the EIW block. In each clock cycle, one of the element-by-element multiplications in (25) is completed by the multiplier with a proper model parameter fed from the EIW block. After each 128 cycles, one of the transformed data is completed and fed into EIW block.

After the eigenspace transformation is completed, the signal components are kept in the $k_{ss} + 1$ registers in the EIW block. Among them, the $(k_{ss} + 1)$th register holds the absolute value of bias for input weighting multiplication. In each clock cycle, the EIW block can compute the output of one of the hidden neurons with the random input weight generated by the LFSR and the sigmoid unit. At the same time, the shared multiplier can perform the output weighting multiplication in a pipeline manner by using the result from the EIW block and the proper output weight fed from the $\beta$ memory buffers.

When the computations of an entire one ELM model are finished, the DCU makes the decision whether to continue the computation of another ELM model or let the ensemble output unit calculate the final classification result. The proposed engine outputs the final classification result if the computations for all the classifiers in the ensemble model are done. Then, it can start to collect another compressed ECG signal or update the model parameters if needed.

### D. Fixed-Point Analysis

First, we performed the fixed-point analysis to investigate the trade-off between the achieved accuracy and the word lengths of each register in this engine. Since the multiplier is reused for eigenspace transformation and output weighting multiplication, the bit widths for input compressed data $\hat{x}$ and the output of the sigmoid $h$ should be the same. Also, the bit widths for the transformation matrix $T_{SE}$ and output weight matrix $\hat{\beta}$ should be the same. To determine the bit width of $\hat{x}$, $h$, the signal component in the eigenspace $s_{ss}T_{SE}$, $\beta$, the input of sigmoid $z$, and the bias $b$, the performance of SE-ELM framework is explored to evaluate the inference accuracy with different fixed-point precisions. The bit widths of $T_{SE}$ and $\hat{\beta}$ have the most significant effect on the inference accuracy, as shown in Fig. 12. After the exhausted computer simulation, the bit widths of $\hat{x}$, $h$, $s_{ss}T_{SE}$, $\beta$, $z$, and $b$ are set to 8, 8, 9, 9, 13, and 8, respectively.

### E. Implementation Results and Comparisons

To validate the function of the proposed architecture design, we have implemented the proposed engine in TSMC 90 nm technology by using Cadence SoC encounter tool. As shown in Fig. 13, the core area of this engine is $384 \mu m \times 384 \mu m$ at 5 MHz operation frequency. Some important characteristics are summarized in Table IX.

Table X shows the comparisons of the proposed AdaBoost-based SE-ELM engine with other recently reported machine learning engines. To make a fair comparison, we normalized the energy consumption and area to 90 nm process. Since the design goal in the on-CS-sensor-analysis is area- and energy-efficient, we defined the area-energy-eficiency ($AEE$, in $mac/pJ \times mm^2$) as

$$AEE = \frac{1}{\text{Energy Efficiency} \times \text{Area}},$$

which describes the computation ability provided by normalized energy consumption and silicon cost.

Since [25] and [30] are both SVM engines that need much more computations for feature extraction or multiplying the support vectors, the energy consumption for each classification is about hundreds of microjoules, which is significantly large when compared to ELM engines. The design in [30] is an SoC based machine learning accelerator that includes a low-power CPU core for feature extraction, thus the area and energy consumption are much larger, as shown in Table X. As for the recently published ELM engines, [7], [13], [31] utilized the mismatches in current mirrors to perform effective random matrix multiplication in the first layer of ELM. Reference [7] targets on neural decoding that needs to achieve the goal.

**Fig. 12.** Inference accuracy of the proposed ensemble of SE-ELM in the worst case with different fixed-point precisions.

**Fig. 13.** Layout of the proposed ensemble of SE-ELM engine.

**Table IX**

| Proposed Ensemble of SE-ELM Engine | | |
|-----------------------------------|---------------------------------|
| Application                       | AF detection by compressed ECG signals |
| Network specification             | Input dimension=128, Input neurons=29, Hidden neurons=400, Output neurons=1 |
| Technology                        | TSMC 90nm 1P9M CMOS Process |
| Voltage                           | 0.9 V |
| Core area                         | $0.1475 \text{ mm}^2 (384 \mu m \times 384 \mu m)$ |
| Frequency                         | 5 MHz |
of extreme low-power design, while [13], [31] are more energy efficient due to the faster operation. In addition, [31] implemented the dimension expansion and physical unclonable function (PUF) for general Internet-of-Things applications. However, they need one capacitor for each row in the current mirror to control the noise issue, which results in larger chip area in these designs. As shown in Table X, [13], [31] show very good energy efficiency, but their areas are much larger than the digital designs. More importantly, the randomness of the first layer of ELM in these works stems from the fabrication mismatch. Therefore, these designs are hard to extend to ensemble design. Note that, the reported area only considers the mixed-signal chip for the input layer of ELM. The output layer of ELM is implemented off chip on an FPGA. In addition, the temperature dependence of analog design also results in inference accuracy degradation.

In our design, we need accurate computations for the eigenspace transformation. Therefore, we choose the full digital design. A recently published work [32] also proposed the VLSI implementation of ensemble ELM and usage of random number generator. In [32], each element in the input weight matrix is represented by multiple bits. Therefore, given an \( n \)-dimension input vector, the multiplication of each hidden neuron needs a multiplier executing for \( n \) cycles. Our proposed design only needs one cycle without any multiplier for the input weighting multiplication by using random Bernoulli matrix as input weight matrix. With the proposed hardware-friendly algorithm and hardware sharing architecture design, the proposed engine can provide a competitive AEE compared with the previous designs that only implemented the first stage of ELM, as shown in Table X. Furthermore, our design can provide the functionality of ensemble method that can improve the inference accuracy of ELM models at the cost of energy efficiency due to eigenspace transformation.

VI. CONCLUSION

In this work, we propose a robust and lightweight ensemble of ELM based on eigenspace transformation and hardware-friendly architecture design. The proposed E-ELM can improve the learnability of compressed noisy signals and thus reduce the required number of ELM classifiers for lightweight model. The proposed SE-ELM further removes the interference directly in the compressed domain, and makes the lightweight learning model robust against interference. This framework achieves the highest accuracy with lowest computation under white noise and interference environment. In the ASIC design, the implementation results show that the proposed engine can provide competitive area- and energy-efficiency to achieve on-CS-sensor-analysis.

REFERENCES


Ching-Yao Chou (S’16) received the B.S. degree in electrical engineering and the Ph.D. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2014 and 2019, respectively. His research focuses on low-complexity signal processing for edge analysis, trying to link compressive sensing with machine learning.

Sheng-Hui Wang (S’18) received the B.S. degree in electrical engineering and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2016 and 2018, respectively. His research interests include the architecture and algorithm design of error-resistant system, biomedical signal processing, and affective computing.

An-You (Andy) Wu (M’96–SM’12–F’15) received the B.S. degree from National Taiwan University (NTU), Taipei, Taiwan, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, USA, in 1992 and 1995, respectively. In 2000, he joined the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering (GIEE), NTU, as a Faculty Member, where he is currently a Distinguished Professor and has also been the Director of GIEE since 2016. From 2007 to 2009, he was on leave from NTU and served as the Deputy General Director of the SoC Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan. His research interests include VLSI architectures for signal processing and communications and adaptive/multirate signal processing. He has published more than 250 refereed journal and conference articles in above research areas, together with five book chapters and 20 granted U.S. patents. He was elevated to IEEE Fellow for his contributions to DSP algorithms and VLSI designs for communication IC/SoC in 2015. He received the Outstanding EE Professor Award from The Chinese Institute of Electrical Engineering, Taiwan, in 2010. He is currently a Board of Governor Member of the IEEE Circuits and Systems Society from 2016 to 2018.