SensorNet: A Scalable and Low-Power Deep Convolutional Neural Network for Multimodal Data Classification

Ali Jafari, *Student Member, IEEE*, Ashwinkumar Ganesan, Chetan Sai Kumar Thalisetty, Varun Sivasubramanian, Tim Oates, *Member, IEEE*, and Tinoosh Mohsenin, *Member, IEEE*

***Abstract*—This paper presents SensorNet which is a scalable and low-power embedded deep convolutional neural network (DCNN), designed to classify multimodal time series signals. Time series signals generated by different sensor modalities with different sampling rates are first converted to images (2-D signals), and then DCNN is utilized to automatically learn shared features in the images and perform the classification. SensorNet: 1) is scalable as it can process different types of time series data with variety of input channels and sampling rates; 2) does not need to employ separate signal processing techniques for processing the data generated by each sensor modality; 3) does not require expert knowledge for extracting features for each sensor data; 4) makes it easy and fast to adapt to new sensor modalities with a different sampling rate; 5) achieves very high detection accuracy for different case studies; and 6) has a very efficient architecture which makes it suitable to be deployed at Internet of Things and wearable devices. A custom low-power hardware architecture is also designed for the efficient deployment of SensorNet at embedded real-time systems. SensorNet performance is evaluated using three different case studies including physical activity monitoring, stand-alone tongue drive system, and stress detection, and it achieves an average detection accuracy of 98%, 96.2%, and 94% for each case study, respectively. We implement SensorNet using our custom hardware architecture on Xilinx FPGA (Artix-7) which on average consumes 246-μJ energy. To further reduce the power consumption, SensorNet is implemented using application-specified integrated circuit at the post layout level in 65-nm CMOS technology which consumes approximately 8× lower power compared to the FPGA implementation. In addition, SensorNet is implemented on NVIDIA Jetson TX2 SoC (CPU + GPU) and compared to TX2 single-core CPU and GPU implementations, FPGA-based SensorNet obtains 15× and 4× improvement in energy consumption.**

***Index Terms*—Multimodal time series, deep neural networks, energy efficiency, FPGA, low power, embedded systems, classification.**

Manuscript received January 10, 2018; revised April 15, 2018 and May 30, 2018; accepted June 1, 2018. This work was supported by the National Science Foundation under Grant 00010145. This paper was recommended by Associate Editor P. K. Meher. *(Corresponding author: Ali Jafari.)*

The authors are with the Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250 USA (e-mail: a137@umbc.edu; gashwin1@umbc.edu; tchetan1@umbc.edu; varuns1@umbc.edu; oates@cs.umbc.edu; tinoosh@umbc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2018.2848647

I. Introduction

TIME series data is a sequence of data points in time order, that is gathered in different kinds of domains from healthcare [1], [2] where one can track a patient's vital signs (heart rate, blood pressure), to fitness and wellness where one can monitor a person's activity [3], to engines in cars and power plants using sensors. Hence, modeling and classifying time series has a wide range of applications. All these datasets are represented by a time series which is univariate or multivariate (multimodal) depending on the number of sensor modalities being measured. Multimodal signals are generated by different sensors usually with different sampling frequencies such as accelerometers, magnetometers, gyroscopes and heart rate monitors. Multimodal devices can increase the number of alternatives available to users to perform different tasks simultaneously.

Traditionally, time series classification problems have been solved with approaches like Dynamic Time Warping (DTW) [4] and k-nearest neighbor (k-NN) [5]. These methods or a combination of them provide a benchmark for current time series classification research [6]. However, there are some challenges with applying these methods: 1) different signal processing techniques such as feature extraction and classification are needed to process the data generated by each sensor modality, which can lead to a long design time, 2) requires expert knowledge in designing the features, and 3) is unscalable when adding new sensor modalities.

Recently, Deep Neural Networks (DNN) have become popular for multimodal time series signals processing [7]–[16]. However, DNN solutions usually have large and power-hungry architectures which is not suitable for deployment at Internet of Things (IoT) and wearable devices.

In this paper, Sensornet shown in Fig. 1 is proposed which is a scalable Deep Convolutional Neural Network (DCNN) designed to classify multimodal time series signals in embedded, resource-bound settings that have strict power and area budgets. SensorNet: (1) is scalable as it can process different types of time series data with variety of input channels and sampling rates. (2) does not need to employ a separate signal processing techniques for processing the data generated by each sensor modality. (3) does not require expert knowledge for extracting features for each sensor data. (4) achieves very high detection accuracy for different case studies. (5) has

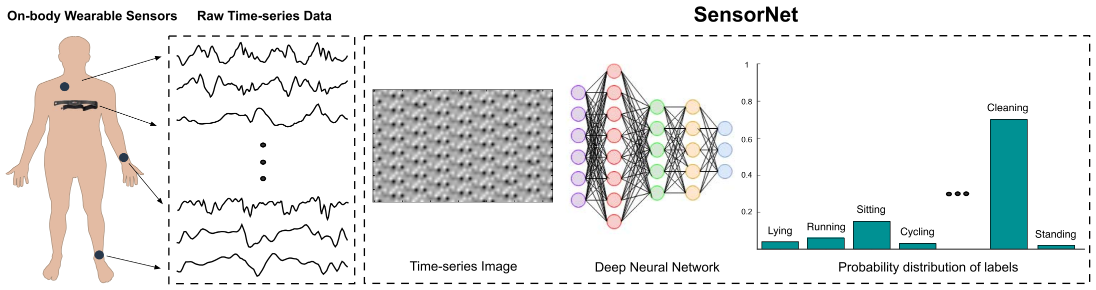

Fig. 1. High-level diagram of the proposed Sensornet.

a very efficient architecture which makes it suitable to be deployed at low power and resource-bound embedded devices.

This paper makes the following major contributions:

- Propose SensorNet, which is a scalable Deep Convolutional Neural Network for multimodal signals classification
- Perform extensive hyperparameter optimization for SensorNet with the goal of reducing power consumption and memory requirements while achieving high detection accuracy
- Evaluate SensorNet performance in terms of detection accuracy, memory requirements and number of computations using three real-world case studies including Physical Activity Monitoring, stand-alone dual-mode Tongue Drive System and Stress Detection
- Propose a custom low power hardware architecture for efficient deployment of SensorNet at IoT and wearable devices, which can perform the entire signal processing in real-time
- Implement SensorNet on FPGA and post layout ASIC in 65-nm CMOS technology and provide results and analysis in terms of power consumption, latency and resource utilization for all three case studies
- Implement SensorNet on embedded NVIDIA Jetson SoC TX2 commercial platform and provide comparisons with FPGA and ASIC results

The rest of this paper is organized as follows: Related prior work is provided in Section II. Section III describes the proposed SensorNet architecture design. Section IV provides the experimental results and analysis. Section V explores different hyperparameters optimization. Hardware architecture design is discussed in Section VI. Hardware implementation results are provided in Section VII. Finally, we conclude in Section VIII.

II. Related Work

In recent years, several multimodal data classification approaches have been proposed which are discussed in this section. Wang and Oates [9] first converted time series into images and then used CNN for processing. The authors converted the time series into an image using two types of representations, i.e., Grammian Angular Fields (GAF) and Markov Transition Fields (MTF). The above mentioned architectures either modeled each variable separately before correlating them or required preprocessing the stream into an image. Zheng *et al.* [8] proposed an architecture which employs a CNN per modality (variable) that processes each variable separately and then correlates them using a fully (dense) connected layer. They tested their network on two different datasets including Physical Activity Monitoring and Congestive Heart Failure Detection and they achieved a detection accuracy of 93% and 94%, respectively. Ordóñez and Roggen [10] proposed a generic deep framework for activity recognition based on convolutional and LSTM recurrent units and evaluated their framework on the Opportunity and the Skoda datasets. Vepakomma *et al.* [11] proposed A-Wristocracy, a Deep Learning Neural Network-based framework for recognizing in-home activities of human users with wrist-worn device sensing. They validated 22 daily activities and achieved 90% test accuracy. Their network consists two hidden layers and is designed specifically for their sensing system. Li *et al.* [12] introduced a system that recognizes concurrent activities from real-world data captured by multiple sensors of different types using 7 layers CNN that extracts spatial features, followed by a long-short term memory network that extracts temporal information in the sensory data. They tested their system with three datasets. Their proposed network has 27M model weights which requires a large memory for saving on an embedded system. Yao *et al.* [13] proposed DeepSense which is a deep learning framework for time series mobile sensing data processing. DeepSense integrates convolutional and RNN to exploit and merge local interactions among similar mobile sensors and extract temporal relationships to model signal dynamics. They deployed DeepSense at Nexus5 and Intel Edison and reported latency and power consumption results. Guan and Ploetz [14] proposed Ensembles of deep LSTM learners for Activity Recognition using Wearables. They tested their approach on three different datasets including Opportunity, PAMAP2 and Skoda. Jiang and Yin [15] proposed an approach that assemble signal sequences of accelerometers and gyroscopes into a novel activity image, which enables Deep CNN to automatically learn the features from the activity image for the activity recognition task. 2D Discrete Fourier Transform is applied to the signal image and its magnitude is chosen as their activity image. Rajpurkar *et al.* [16] proposed a cardiologist-level arrhythmia detection using a 34-layer CNN and they exceed the average cardiologist performance in both sensitivity and precision. Basterretxea *et al.* [17] proposed a solution for multimodal activity recognition on FPGA, which uses different signal processing algorithms including feature extraction, Principle Component Analysis (PCA) and a 2-layer Neural Network.

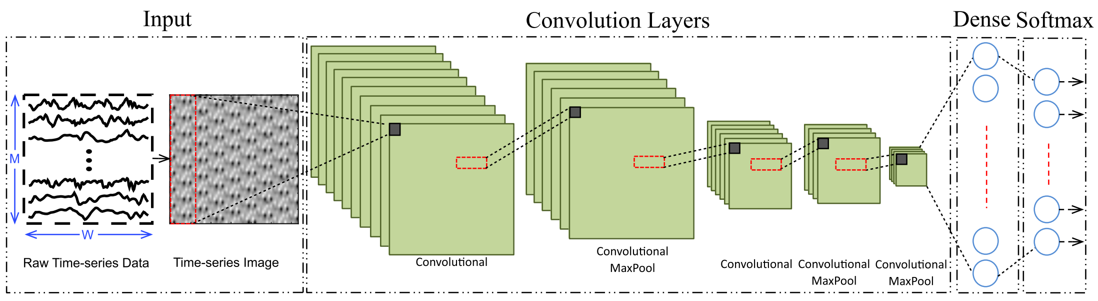

Fig. 2. The proposed Sensornet architecture which consists of different kinds of layers. The network takes a 1-channel image (constructed from the raw sensor signals) as input and consists of 5 convolutional layers, 3 pooling layers, 1 dense layer and a softmax layer (for prediction). The architecture is denoted by 32(Mx5)-16(1×5)-2-16(1×5)-8(1×5)-2-8(1×5)-2-64-N and its generic notation is C_1(size)-C_2(size)-S_1-C_3(size)-C_4(size)-S_2-C_5(size)-S_3-H_1-N.$C_1..C_5$ are the convolutional layers, $S_1..S_3$ are the max-pooling layers, H_1 is the size of dense layer and N is the softmax layer of size N.

In summary, most of the previous work do not propose a real-time hardware solution for multimodal time-series data classification or use general-purpose processors [13], [18] which results in high power consumption. Also, the architecture developed in previous work are not efficient for hardware deployment at IoT and wearable devices [8], [14]. Furthermore, some of the previous related work require expert knowledge in designing the features which results in a long design time [19] and those work are not scalable when adding new sensor modalities.

III. SensorNet Architecture Design

An overview of the proposed SensorNet architecture is discussed here. SensorNet is designed to capture correlations between various modalities simultaneously. To capture these correlations, depends on the application a snapshot of raw signals with a fixed window size is converted to an image (2-D signal) and is passed to the network and is processed. Fig. 2 shows a high-level block diagram of the proposed system which consists of preprocessing, convolutional, fully connected and softmax layers.

A. Signal Preprocessing

Consider a given time series that consists of M modalities/variables with same or different sampling frequency. Prior to training, each variable is independently normalized using the $l2$ norm. To generate an image from the normalized variables, a sliding window of size W and step-size S is passed through all variables, creating a set of images of shape $1 \times W \times M$ (single channel image). The label associated with this image depends on the dataset. The datasets used to test the network in this paper contain a label for every time step. Since a single label is assigned to each image, the label of the current time step is taken as the label of the image (and the label that needs to be predicted subsequently while testing). A given image generated at time-step I_t has the prior states of each variable from $(t - W + 1) \dots t$. Thus, the network can look back W prior states of each variable and given the current state of each variable, predicts the label.

B. Neural Network Architecture

Fig. 2 shows SensorNet architecture. It consists of 5 convolutional layers, 1 fully connected and a softmax layer that is equivalent in size to the number of class labels (depending on the case study). In the pre-processing stage, SensorNet takes the input time series data and fuses them into images. Then, the images are passed into the convolutional layers and some features which are shared across multiple modalities are generated using a set of local filters. Then, these features are fed into the fully-connected and the softmax layers. SensorNet architecture including number of layers, number of filters and filter shapes for each layer are chosen based on an extensive hyperparameter optimization process which will be discussed in details in Section V.

First, second, third, fourth and fifth convolutional layers contain 32, 16, 16, 8 and 8 filter sets, respectively. The convolution filters have a height of either M or 1, because it's assumed that there are no spatial correlations between the variables. Also, the ordering of variables prior to generating images doesn't affect the ability of the network to perform classification. A filter of height M or 1 remains unaffected by the ordering of the variables. Therefore, the filter size for the first convolutional layer is $M \times 5$ where M is number of input modalities. For other layers, filter shape of 1×5 is chosen.

Max-pooling is applied thrice, once after the second convolutional layer, then after the fourth convolutional layer and the last one after the fifth convolutional layer. A max-norm regularization of 1 is used to constrain the final activation output. The pooling size for all max-pooling layers is 1×2. Once the convolution operations have been performed, the image is flattened into a single vector so that a fully connected layer can be added.

Two fully connected layers are employed in SensorNet which the first one has a size of 64 nodes and the second one has a size equivalent to the number of class labels with Softmax activation. All the layers of the network have their weights initialized from a normal distribution. A learning rate of 0.0001 is used to train the network. Rectified Linear Unit (ReLU) is used as activation functions for all the layers. The network is trained using backpropagation and optimized using RMSprop [20]. Categorical crossentropy is used as the loss function. Following is the loss function:

$$L(y_p, y_a) = \frac{-1}{N} * \sum_{i=1}^{N} [y_a^i log y_p^i - (1 - y_a^i) log(1 - y_p^i)] \quad (1)$$

where, y_p is the predicted label and y_a is the expected label.

As shown in algorithm 1, there are three main functions defined:

TABLE I
INFORMATION FOR THREE DIFFERENT CASE STUDIES INCLUDING PHYSICAL ACTIVITY MONITORING, sdTDS AND STRESS DETECTION

Application	# of Activity labels	Sensors Position	Sampling Rate (Hz)	# of Subjects	# of Channels
Physical Activity	12	Chest & Arm & Ankle	100 & 9	8	40
Tongue Drive	12	Headset	50	2	24
Stress Detection	4	Wrist & Finger	8 & 1	20	7

Algorithm 1 Train *SensorNet* to Predict Labels (Actions)

```
Input: The Network $N$ as defined in Figure 2. An input
       dataset $D$ of size $k$ $(d_1..d_k)$ sampled from various
       sensors with each point having $M$ attributes.
Output: Predict the class label $l_i$ for a datapoint $d_i$
# Consider the training batch size to be b, the learning
rate $LR$ and reshape() changes the shape of the tensor.
# W is the size of the sliding window.
# epochs is the number of epochs for which the model is
trained.
# For categorical crossentropy refer to Eq.1
# $x_{train}$ is a list of images and $y_{train}$ has the expected
labels.
for $i \leftarrow W$ to $D$ do
    # After reshape the tensor is of shape (1, W, M)
    $x^i_{train}$ = reshape($d_{i-W+1}..d_i$)
    $y^i_{train} = l_i$
# Train the model.
for $e \leftarrow 1$ to epochs do
    for $j \leftarrow 1$ to $\lfloor \frac{D}{b} \rfloor$ do
        batch = $x_{train}[j * b : (j+1) * b]$
        $y_{pred}$ = forwardpass(N, batch)
        loss = L($y_{pred}$, $y_{train}[j * b : (j+1) * b]$)
        g = backwardpass(loss)
        gradientupdate(g)
return $N$
```

- **Reshape:** This function reshapes a given tensor into another form. We transform a 2D matrix to a 3D tensor with the first axis as 1 representing a single channel.
- **Forwardpass:** It is a single complete processing of the input image to predict the label (for the given dataset).
- **Backwardpass:** It computes the gradient of weights with respect to the loss function (required to perform gradient descent).
- **GradientUpdate:** This function updates the weights using the gradient and the defined learning rate.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, SensorNet is evaluated using three real-world case studies including Physical Activity Monitoring [21], stand-alone dual-mode Tongue Drive System (sdTDS) [22] and Stress Detection [23] and in-depth analysis and experimental results are provided.

The information for all the case studies are shown in Table I. As it can be seen from the Table, the sampling rates of the sensors for each case study are different in range of 1 Hz to 100 Hz. Also, the sensors are placed in variety of spots on human body including chest, arm, ankle, head and hand fingers. The number of channels for each case study refers to the number of input time series signal which are received simultaneously by SensorNet. Physical Activity Monitoring, sdTDS and Stress Detection case studies can be considered to generate large, medium and small size datasets.

For all the case studies, SensorNet is trained using Keras [24] with the TensorFlow as backend on a NVIDIA 1070 GPU with 1664 cores, 1050 MHz clock speed and 8 GB RAM. Models are trained in a fully-supervised way, backpropagating the gradients from the softmax layer to the convolutional layers.

A. Case Study 1: Physical Activity Monitoring

1) Dataset: Physical Activity Monitoring dataset (PAMAP2) [21] records 12 physical activities performed by 9 subjects. The physical activities are, for instance: 'standing', 'walking', 'lying' and 'sitting'. Three IMUs (inertial measurement units) and one heart rate monitor are placed on chest, arm and ankle to record the data. The sampling frequency of the IMU sensors is 100 Hz and the heart rate monitor sensor has a sampling frequency of 9 Hz. In total, the dataset includes 52 channels of data but 40 channels are valid according to [21]. Also, out of 9 subjects the data of 8 subjects are used, as subject 9 has a very small number of samples.

2) Experiment Setup and Results: As we mentioned in Section III, SensorNet utilizes 5 convolutional layers, followed by 2 fully-connected layers. First convolutional layer has 32 filter sets and each filter size is 40×5. Other convolutional layers have 16, 16, 8 and 8 filter sets with a size of 1×5. For this experiment, 80%, 10% and 10% of the entire data for each subject is chosen randomly as the training, validation and testing set, respectively. To determine the number of required epochs for the training, we train SensorNet for 150 epochs and plot validation and training loss and accuracy results. As is shown in Fig. 3, after 100 epochs the validation loss and accuracy are stable and satisfactory. Therefore, for all the experiments for this dataset we train SensorNet with 100 epochs.

After training SensorNet, we evaluate the trained model to determine the detection accuracy. Fig. 4 shows the classification accuracy of SensorNet for the Physical Activity Monitoring case study for different subjects with a sliding window of size 64 samples and step-size of 1-16-32-64. As can be seen from the figure, all subjects with step-size 1 achieve a high detection accuracy. However, as the step-size increases from 1 to 64 the detection accuracy decreases. The average

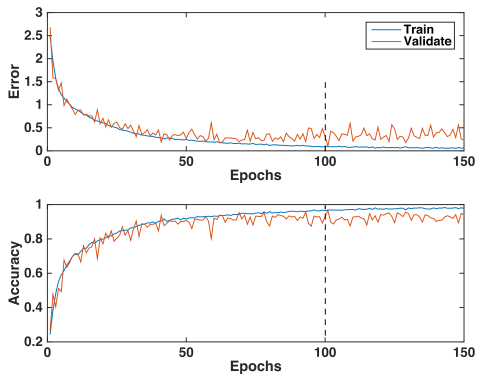

Fig. 3. Error and accuracy of the training and validation sets for Physical Activity Monitoring case study over 150 epochs. The vertical dashed line indicates the determined epoch.

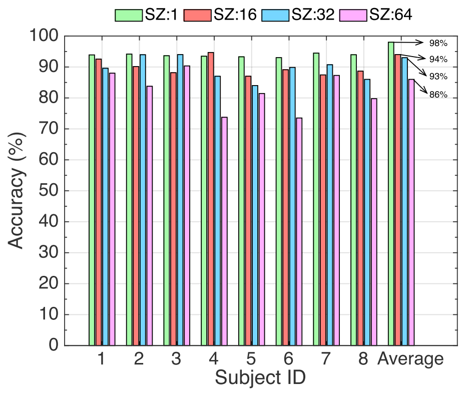

Fig. 4. Comparison of SensorNet classification accuracy for Physical Activity Monitoring case study. The results are for different subjects with a sliding window of size 64 samples and step-size (SZ) of 1-16-32-64.

accuracy of all subjects with step-sizes of 1, 16, 32 and 64 are 98%, 94%, 93% and 86%, respectively.

B. Case Study 2: Stand-Alone Dual-Mode Tongue Drive System (sdTDS)

1) sdTDS Overview and Experimental Setup: In [22], we proposed and developed stand-alone Tongue Drive System (sTDS) which is a wireless wearable headset and individuals with severe disabilities can use it to potentially control their environment such as computer, smartphone and wheelchair using their voluntary tongue movements [25]. In this work, in order to expand the functionality of sTDS, we propose a stand-alone dual-mode Tongue Drive System (sdTDS) by adding head movements detection to the previous version. Fig. 5 shows sdTDS prototype which includes a local processor, four magnetic and acceleration sensors, a BLE transceiver, a battery and a magnetic tracer which is glued to the user's tongue. Two magnetic and acceleration sensors are placed on each side of the headset and the processor is placed in a box at backside of the headset. The box is also used for placing a battery and it's weight is around 0.14 lb. The box is designed using 3D printing technology. In order to generate user-defined commands, user should move his/her tongue to 7 specific teeth or move his/her head to 5 different directions. The raw

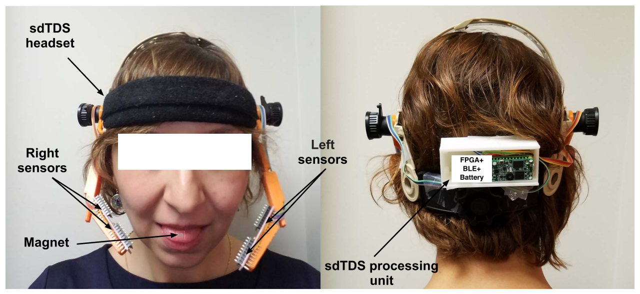

Fig. 5. sdTDS prototype placed on a headset which includes a FPGA, four acceleration and magnetic sensors, a Bluetooth low energy transceiver, a battery and a magnetic tracer which is glued to the user's tongue.

data generated by 4 magnetometers and accelerometers are transferred into a FPGA processor where the entire signal processing including feature extraction and classification is performed by SensorNet and 12 different user-defined commands can be generated.

2) Experiment Results: Several different data sets are captured using sdTDS for training and testing purpose. sdTDS generates 24 channels of time series data that corresponds to tongue and head movements. As it was mentioned in Section III, SensorNet utilizes 5 convolutional layers, followed by 2 fully-connected layers. For the sdTDS, first convolutional layer has 32 filter banks and each filter size is 24×5. Other convolutional layers have 16, 16, 8 and 8 filter banks with a size of 1×5. For this experiment, 80%, 10% and 10% of the entire data for each trivial is chosen randomly as the training, validation and testing sets, respectively. We train SensorNet for 100 epochs.

After training SensorNet using sdTDS dataset, we evaluate the trained model to determine the detection accuracy. Based on previous experiments, we train and test the sdTDS with a sliding window of size 64 samples and step-size of 1, as the step-size of 1 gives better detection accuracy, consistently. For sdTDS case study, SensorNet detection accuracy for tongue and head movements detection is approximately 96.2%.

C. Case Study 3: Stress Detection

1) Dataset: This database contains non-EEG physiological signals used to infer the neurological status including physical stress, cognitive stress, emotional stress and relaxation of 20 subjects. The dataset was collected using non-invasive wrist worn biosensors. A wrist worn Affectiva collects electrodermal activity (EDA), temperature and acceleration (3D); and a Nonin 3150 wireless wristOx2 collects heart rate (HR) and arterial oxygen level (SpO2) data [23]. Therefore, in total the dataset includes 7 channels of data. The sampling frequency of wrist worn Affectiva is 8 Hz and wristOx2 has a sampling frequency of 1 Hz.

2) Experiment Setup and Results: As it was discussed in Section III, SensorNet utilizes 5 convolutional layers, followed by 2 fully-connected layers. First convolutional layer has 32 filter sets and each filter size is 7×5. Other convolutional layers have 16, 16, 8 and 8 filter sets with a size of 1×5. Similar to Physical Activity Monitoring case study, for this

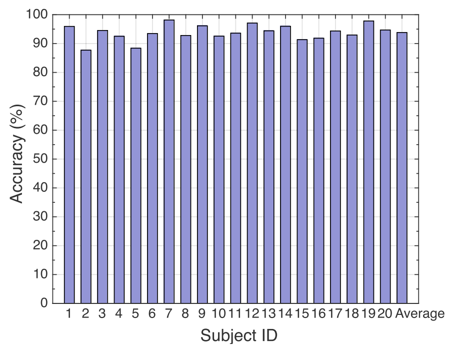

Fig. 6. Comparison of SensorNet classification accuracy for Stress Detection case study. The results are for different subjects with a sliding window of size 64 samples and step-size of 1.

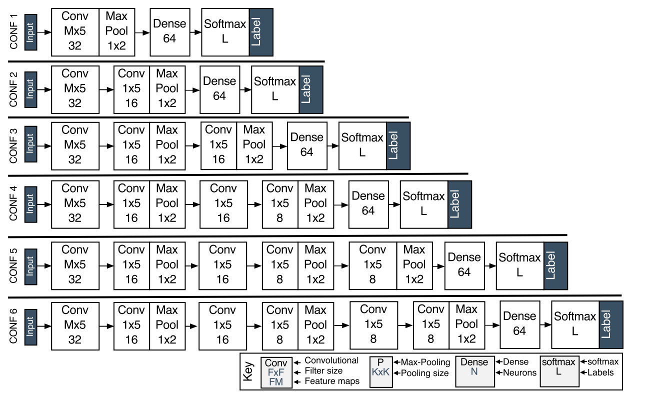

Fig. 7. Comparison of six different SensorNet configurations. M and L are the number of input data channels and labels for different case studies, respectively.

experiment 80%, 10% and 10% of the entire data for each subject is chosen randomly as the training, validation and testing set, respectively. To determine the number of required epochs for the training, we train SensorNet for 150 epochs and plot validation and training loss and accuracy results. After 100 epochs the validation loss and accuracy are stable and satisfactory. Therefore, for all the experiments for this dataset we train SensorNet with 100 epochs. Fig. 6 shows the classification accuracy of SensorNet for Stress Detection case study for 20 different subjects. As is shown in the figure, most of the subjects have a high detection accuracy more than 90%. The average accuracy of all 20 subjects is approximately 94%.

V. SensorNet Optimization and Complexity Reduction

As it was discussed in section III, SensorNet utilizes 5 convolutional layers, followed by 2 fully-connected layers. First convolutional layer has 32 filter sets and each filter size is $M \times 5$, where M is the number of input channels. Other convolutional layers have 16, 16, 8 and 8 filter banks with a size of 1×5. The first fully-connected layer has 64 nodes and the number of nodes in the last one is equivalent to the number of labels for any specific application. In this section, we explain the reason behind of choosing SensorNet architecture and parameters.

One of the primary objectives of this paper is to be able to efficiently deploy SensorNet at IoT and wearable devices which have strict power and area budgets. Therefore, we perform extensive hyperparameter optimization for SensorNet with the goal of reducing memory requirements, hardware complexity and power consumption while achieving high detection accuracy. In this section, we specifically explore the impact of changing the following parameters or configurations on SensorNet performance: *1) Number of convolutional layers, 2) Number of filters, 3) Filter shapes, 4) Input zero-padding, and 5) Activation functions.*

A. Number of Convolutional Layers

In this experiment, we compare six SensorNet configurations with an increasing number of convolutional layers, for the three different case studies. These 6 configurations are depicted in Fig. 7. The comparison has been made in terms of detection accuracy, number of convolutional operations, number of parameters (model weights) and memory requirements. Fig. 8-A, 8-B, 8-C and 8-D depict the impact of increasing the number of convolutional layers on the number of model parameters and memory requirements. As is shown in the figures, by increasing the number of convolutional layers, the total number of model parameters and memory requirements decrease which is desired. The reason is that we use three max-pooling layers after the convolutional layers, therefore by adding more convolutional layers the size of the time series images shrink and the fully-connected layer needs to process less number of data and thus requires less memory. Fig. 8-E shows the impact of increasing the number of convolutional layers on detection accuracy. As it can be seen from the figure, if the neural network is too shallow high-level features can not be learned, therefore the detection accuracy is low. However, the results show that, by increasing the number of convolutional layers detection accuracy increases but up to 5 convolutional layers. After that, for Activity Monitoring and sdTDS case studies the accuracy improve slightly but for the Stress Detection reduces because the useful features may be filtered out during the convolutional and max-pooling processes. Also, by adding additional convolutional layer the number of operations to finish a classification task increases slightly which is shown in Fig. 8-F. This analysis results show that SensorNet with 5 convolutional layers is the best candidate with regards to detection accuracy, number of convolutional operations and memory requirements.

B. Number of Filters

The number of filters (weights) are another important hyperparameter for implementing SensorNet on low power and resource-limited embedded devices because the number of model weights affect the memory requirements and also the number of required computations to finish a classification task. The number of required computations has a direct effect on energy consumption. In this experiment, we keep the number of convolutional layers fixed (5 layers) and increase the number of filters for each layer as is shown in Fig. 9.

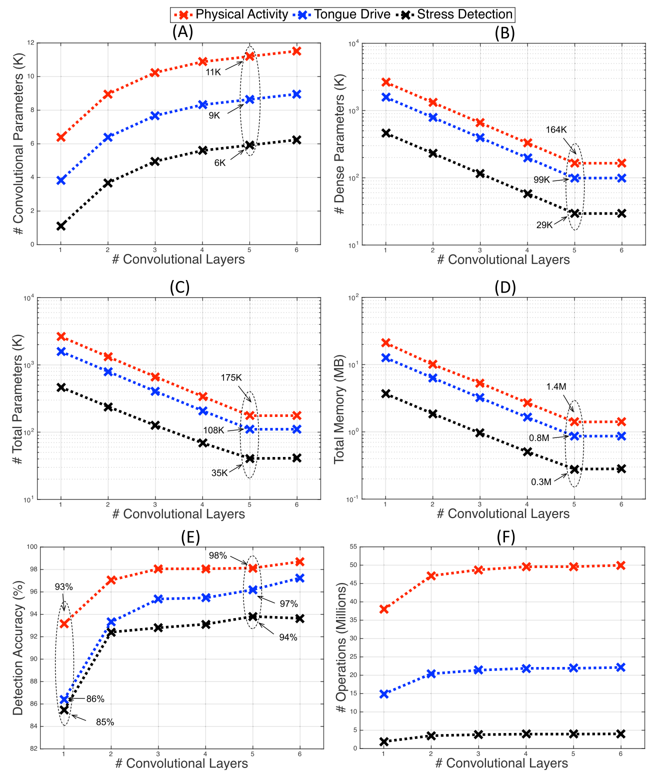

Fig. 8. Impact of increasing number of convolutional layers on memory requirements, detection accuracy and number of operations of SensorNet, for three different case studies.

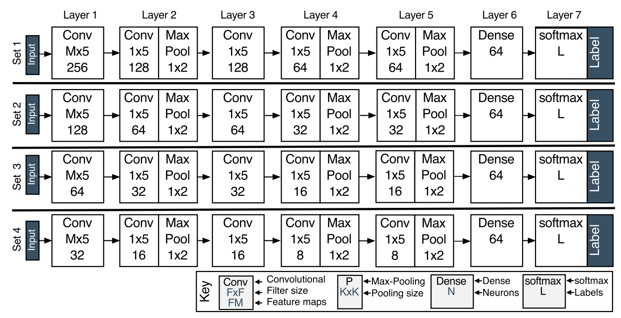

Fig. 9. SensorNet configurations with four different filter sets. Number of filters for convolutional layers are doubled for each filter set. M and L are the number of data channels and labels for different case studies, respectively.

The goal of this experiment is to find the impact of the number of filters on the detection accuracy, number of convolutional operations, number of parameters (model weights) and memory requirements for Physical Activity Monitoring, sdTDS and Stress Detection case studies. Therefore, SensorNet is trained and tested using four different configurations with different number of filter sizes. Fig. 10 shows a comparison of number of required parameters (model weights) for different trained models. Model weights includes the parameters for convolution, fully-connected and softmax layers. As is shown in the figure, as we increase the number of filters for each layer, the detection accuracy improves. However, the number of operations, memory requirements and the number of model parameters increase which is not desire for hardware implementation in a resource limited embedded platform.

Based on the results, SensorNet with different filter sets achieves similar detection accuracies, but Set 4 needs lower number of parameters and requires smaller memory compared to other filter sets and therefore is chosen to be implemented on hardware.

C. Filters Shapes

Another important parameter for implementing SensorNet on low power embedded platforms is the filter shape. As it was explained in Section III the idea is to generate some shared features between different input modalities. Therefore, we choose to have the filters with size $M \times 5$, where M is the number of input modalities, for the first convolutional layer. For other convolutional layers the filters are 1×5. By employing this size of filter without zero padding the outputs of the first layer are 1-D vectors and the following layers also will be 1-D vectors. This will improve the memory requirements on an embedded platform drastically; because the feature maps will be 1-D signal which compared to an image is much smaller. Also, smaller number of model weights are needed as the dense layer takes 1-D vectors rather than images. Furthermore, it reduces the the number of operations which this affects the energy consumption directly, when is implemented on an embedded platform.

In this experiment we change the filters shapes for the first convolutional layer to $M\times5$, 5×5, 3×3 and 1×5 for Physical Activity Monitoring, sdTDS and Stress Detection case studies. Based on the results, filter size of $M\times5$ gives better detection accuracy compared to 5×5, 3×3 and 1×5 filter sizes, as is shown in Fig. 11. Also, another interesting finding is that, for the dataset with more number of input channels, choosing $M\times5$ filter size give better accuracy compared to smaller datasets. Because, the small size filters can cover most of the input channels in the smaller dataset but not in the dataset with many input channels.

D. Zero-Padding

In this experiment we explore the impact of input data zero-padding in the first convolutional layer on detection accuracy, for Physical Activity Monitoring, dTDS and Stress Detection case studies. Input zero-padding makes the output of the convolutional layer to be similar or same as the input to the layer. Based on the results shown in Fig. 12, zero-padding the input data helps with accuracy, although it increases the number of parameters and memory requirement. As it can be seen from the figure, by applying the zero-padding, the detection accuracy increases by 4.6%, 3.4% and 3.8% for Physical Activity Monitoring, sdTDS and Stress Detection case studies, respectively. However, total memory requirements and number of operations are increased 9×, 24×, on average which will affect the power consumption negatively as well. Therefore, we choose to implement SensorNet without zero-padding on the hardware.

E. Activation Functions

In Section III, we mentioned that Rectified Linear Unit (ReLU) activation functions is efficient because it

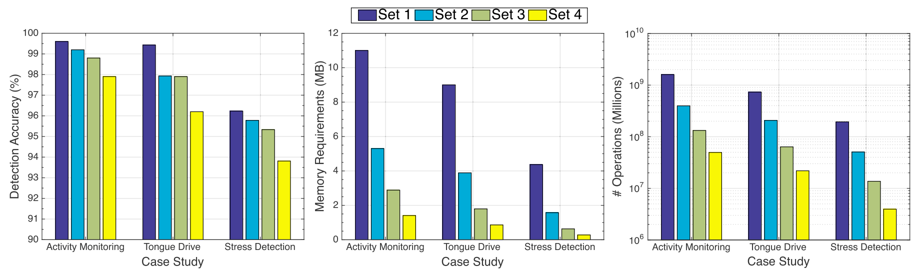

Fig. 10. Comparison of four different SensorNet configurations in terms of detection accuracy, memory requirements and number of operations. By adding additional model weights to each layer, the computation and memory grow dramatically with only modest improvement in detection accuracy.

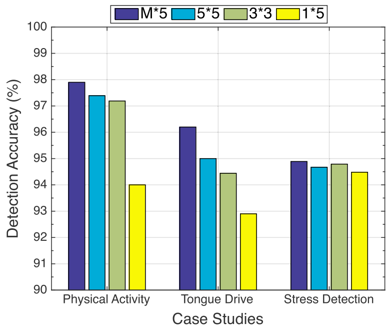

Fig. 11. Comparison of SensorNet detection accuracy for four different filter shapes.

requires few operations to perform. Therefore, it reduces the hardware complexity on a hardware embedded setting. In all the convolutional layers we use ReLU as the activation function. Typically Sigmoid is used as the activation function for the fully-connected layer. However, Sigmoid introduces hardware complexity to the design which is not desired. Thus, in this section, we explore SensorNet detection accuracy by employing different activation functions in the fully-connected layer. In this experiment, we train SensorNet for stand-alone dual-mode Tongue Drive System case study using ReLU as the activation function for all the convolutional layers and using three activation functions including Sigmoid, Tanh and ReLU for the fully-connected layer. The performance results in terms of training accuracy during 100 epochs is shown in Fig. 13. As it can be seen from the figure, SensorNet using any of Sigmoid, Tanh and ReLU activation functions achieves similar accuracies eventually and using different activations function does not affect what SensorNet can learn. Therefore, we choose ReLU as the activation function for all the layers because it has less hardware complexity compared to other activation functions and achieves comparable training and testing accuracy.

VI. SensorNet Hardware Architecture Design

Hardware architecture design for SensorNet faces several challenges such as computational model implementation, efficient parallelism and managing memory transfers. Following are the objective for the hardware architecture design: consumes minimal power, meets latency requirement of an application, occupies small area, needs to be fully reconfigurable and requires low memory. Also, we design SensorNet hardware architecture to be fully reconfigurable, because different applications have different requirements. Parameters such as number of convolutional and fully-connected layers, filter shapes and number of filters are configurable.

Fig. 14 depicts SensorNet hardware architecture with implementation details. This architecture is designed based on Algorithm 1 which was depicted and explained in Section III. The main components of SensorNet on hardware consists of the following:

(A) Convolutional Performs convolutional layer operations. Also, this block includes ReLU activation logic. PE refers to convolution Processing Engine.

(B) Max-pooling Performs max-pooling operations.

(C) Fully-connected Performs fully-connected layer operations. The fully-connected block includes ReLU and softmax activation functions. ReLU will be used as the activation for the first fully-connected layer and softmax will be used for the last fully-connected layer and will perform classification task.

Fig. 14-A shows convolution block. As is shown, convolution block contains one multiplier, one adder/subtractor, one cache for saving filters, input feature maps and output feature maps, a few registers, multiplexers and a state machine block. When the convolution operations are done for all the input feature maps, the output feature maps will be saved into the main feature map memory. The input data coming from the sensors are 16-bit two's complement. Also, the filters are considered to be 16-bit two's complement which will be discussed in sub-section VI-B. After performing the convolution, the data will pass to ReLU activation function. The output of ReLU is truncated to 16 bits and saved in feature map memory. An offline training is performed to obtain model weights using keras. The model weights are converted to fixed-point format and are represented by 14 bits. The floating-point arithmetic is complex, requires more area and consumes more power compared to fixed point arithmetic. Fig. 14-B shows the max-pool block which contains some registers and a comparator. The input to the max-pool is feature maps data, which is formed by convolution block. After max-pooling operations finish, the results will be saved into the main feature map memory. 14-C shows the fully-connected block. As is shown, the architecture consists of a serial dot product engine, a dynamic sorting logic for the softmax activation function, ReLU logic and a state machine for controlling all sub-blocks.

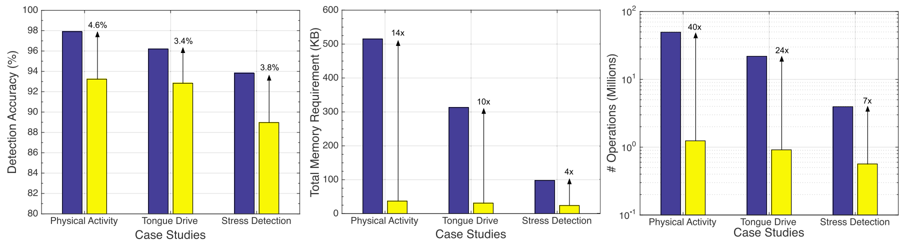

Fig. 12. Impact of zero-padding on SensorNet detection accuracy, memory requirements and number of computations, for Physical Activity Monitoring, sdTDS and Stress Detection case studies.

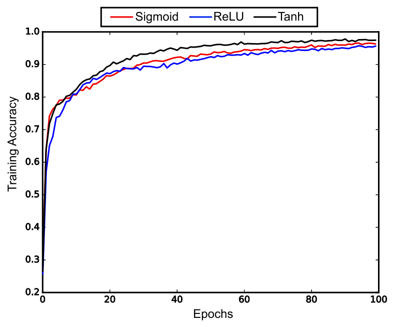

Fig. 13. Impact of different activation functions including Sigmoid, Tanh and ReLU on SensorNet detection accuracy.

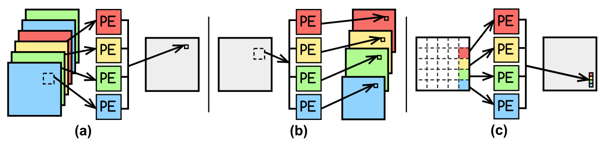

Fig. 15. Comparison of different parallel tiling techniques for convolutional layers. Output channel tiling has the least communication contention and inter-core dependency. (a) Input channel tiling. (b) Output channel tiling. (c) Image patch tiling.

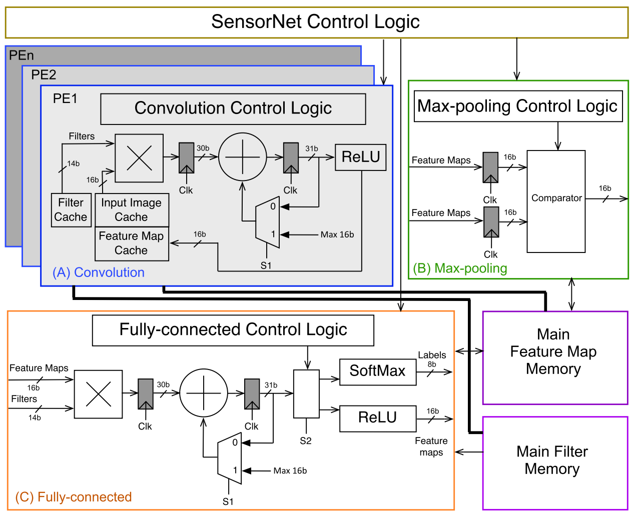

Fig. 14. Block diagram of SensorNet hardware architecture which includes convolution, max-pooling and fully-connected blocks and also a top-level state-machine which controls all the blocks. PE refers to convolution Processing Engine.

Depends on the layer either ReLU or softmax can be used. After finishing computations for the fully-connected layers the results will be saved into the main feature memory.

A. Exploiting Efficient Parallelism

Scalability is one of the key features of the proposed SensorNet on hardware. Therefore, we design SensorNet hardware architecture to be configured to perform convolution operation in parallel if it is needed specially for fast applications. In deep convolutional neural networks, convolutional layers dominate the computation complexity and consequently affects the latency and throughput. Therefore, for the applications with many input modalities or the applications that need to issue a command very fast, we must exploit efficient forms of parallelism that exist within convolutional layers. In [26] we explored three main forms of parallelism methods, that can be employed in convolutional layers. The basic process for the three tiling methods are shown in Fig. 15. The first method, we will refer to as input channel tiling, is to convolve multiple input feature channels concurrently for a given feature map. The second method, output channel tiling, performs convolution across multiple output channels for a given input channel, simultaneously. The third method, which we refer to as image patch tiling, is to break a given input feature channel into patches and perform convolution on the patches concurrently. Zhang *et al.* [27] analyzed these three tiling methods using the rooftop model to determine what method provides the best throughput in FPGA fabric. Their findings confirm that output channel tiling provides the best form of parallelism when taking into account I/O memory bandwidth and computational load using the computation to communication (CTC) ratio. Therefore, we primarily exploit output channel tiling due to minimal dependency among the parallel cores and minimal communication contention.

B. Optimal Fixed-Point Format Width

Another important consideration in SensorNet hardware architecture design is to find optimum level of precision for the weights. This will affect the memory requirements and also the power consumption. We model a custom fixed-point SensorNet design to find optimum weights length. To quantify performance gap between floating point Keras implementations and proposed fixed-point architectures, we calculate average accuracy from the estimated signal obtained from Python software solution and hardware implementations. Fig. 16 shows accuracy of SensorNet computation signal depends on the number of fixed-point bits used to represent the weights.

TABLE II

IMPLEMENTATION RESULTS OF THE PROPOSED SENSORNET ON XILINX FPGA (ARTIX-7). THE RESULTS OBTAINED AT CLOCK FREQUENCY OF 100 MHz

Merits/Case Studies	Physical Activity Monitoring			Tongue Drive System			Stress Detection		
	Serial	Semi Parallel	Fully Parallel	Serial	Semi Parallel	Fully Parallel	Serial	Semi Parallel	Fully Parallel
# of used PE	1	4	8	1	4	8	1	4	8
BRAM	14	23	35	12	18	26	11.5	16	22
# DSP slices	3	6	10	3	6	10	3	6	10
# of Slices	982	1680	2506	961	1592	2430	939	1594	2388
Latency (ms)	14.8	3.8	2	11.2	2.9	1.5	7.3	1.9	1
Throughput (label/s)	67	259	491	89	340	641	136	513	952
Dynamic Power (mW)	46	72	103	42	64	89	41	61	83
Static Power (mW)	71	71	72	71	71	71	71	71	71
Total Power (mW)	116	143	175	113	135	160	112	132	154
Total Energy (mJ)	1.7	0.6	0.35	1.3	0.4	0.24	0.8	0.3	0.15

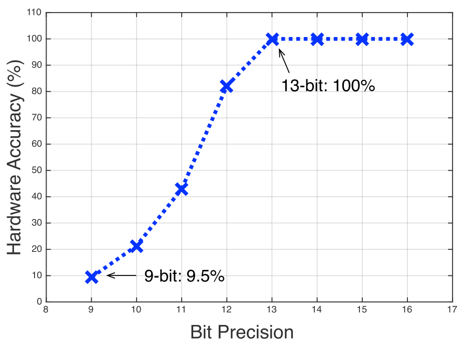

Fig. 16. Hardware accuracy of SensorNet with respect to number of fixed-point bits used to represent the filters (model weights).

TABLE III

SENSORNET ASIC IMPLEMENTATION RESULTS AT OPERATING FREQUENCY OF 100 MHz. CMOS FABRICATION PROCESS IS 65 nm WITH 1 *V* POWER SUPPLY

Metrics/ Case Studies	Physical Activity Monitoring	Tongue Drive System	Stress Detection
Area utilization	95	94	95
Clock freq. (MHz)	100		
Max. clock freq. (MHz)	857	867	888
Core area (mm^2)	0.4	0.3	0.2
Latency (ms)	14.8	11	7
Throughput (label/s)	67	89	136
Leakage power (mW)	7.4	6.3	2.4
Dynamic power (mW)	11.1	9.6	6.8
Total power (mW)	18.5	15.9	9.2
Energy (mJ)	0.27	0.17	0.06

Based on the results, 13-bit precision gives hardware accuracy of 100% with an error of 2^-13. Therefor, all SensorNet filters are converted to 1.13-bit (14-bit) fixed-point format and saved in the main filter memory.

VII. HARDWARE IMPLEMENTATION RESULTS

In this section, SensorNet implementations results on both FPGA and ASIC, for Physical Activity Monitoring, sdTDS and Stress Detection case studies are presented. Also, we provide SensorNet implementations results on NVIDIA TX2 SoC and make a comparison between the results of FPGA, ASIC and TX2 platforms.

A. FPGA Implementation Results and Analysis

The complete proposed SensorNet which includes convolution, max-pooling, fully-connected and activation functions are implemented on an Xilinx Artix-7 FPGA at clock frequency of 100 MHz. Verilog HDL is used to describe SensorNet hardware architecture.

Fig. 17 shows power consumption breakdown of post-place and route implementation on the FPGA, which is obtained by using Vivado Power tool. As it can be seen from the figure, average device static and Block RAMs power consumption of FPGA is around 62% and 18% of entire power which is very large compared to the power consumption of SensorNet logic. However, the overall power consumption for all the case studies is small and is suitable for battery-powered wearable devices.

Table II provides SensorNet performance results for different architectures, including serial, semi-serial and fully-parallel designs. Also, Fig. 18 demonstrates the impact of increasing the number of PEs on power consumption, classification latency and energy. As is shown, for a given network increasing the number of PE, increases power consumption slightly but improves both latency and energy consumption.

B. ASIC Implementation Results and Analysis

To reduce the overall power consumption, an application-specified integrated circuit (ASIC) for SensorNet is implemented at the postlayout level in 65-nm CMOS technology with 1-V power supply. Due to the ability of ASIC designs to run at high clock frequencies, only SensorNet with a serial architecture is implemented, for all the case studies including Physical Activity Monitoring, sdTDS and Stress Detection. A standard-cell register-transfer level (RTL) to Graphic Data System (GDSII) flow using synthesis and automatic place and route is used. The proposed SensorNet including convolution, max-pooling, fully-connected with activation functions is implemented using Verilog to describe the architecture, synthesized with Synopsys Design Compiler, and place and routed using Cadence SOC Encounter. The ASIC layouts for all the case studies are shown in Fig. 19. The implementation results are provided in table III. SensorNet is able to operate at maximum clock frequency of 888 MHz. However, the clock frequency has been reduced to 100 MHz to reduce the power consumption and have a fair comparison with FPGA performance results. The ASIC implementation reduces the power consumption by a factor of 8 on average.

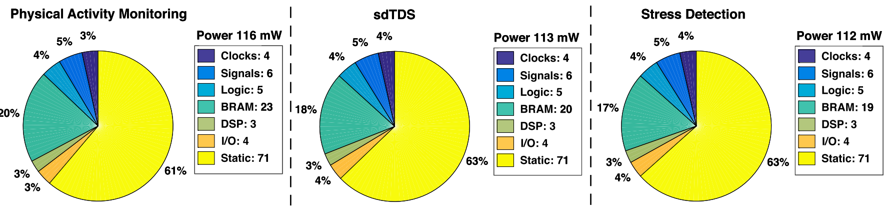

Fig. 17. Serial implementation power consumption results on FPGA, for Physical Activity Monitoring, sdTDS and Stress Detection case studies.

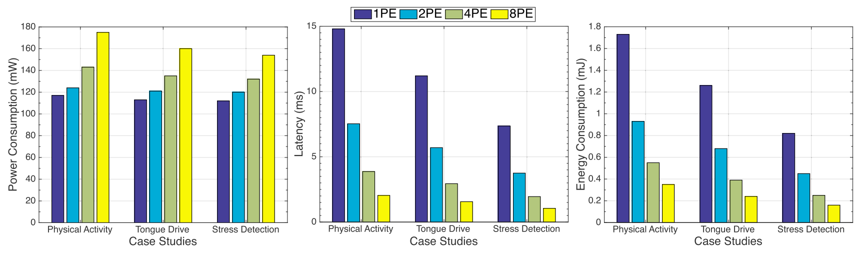

Fig. 18. Impact of number of used PE on power consumption, energy consumption and latency.

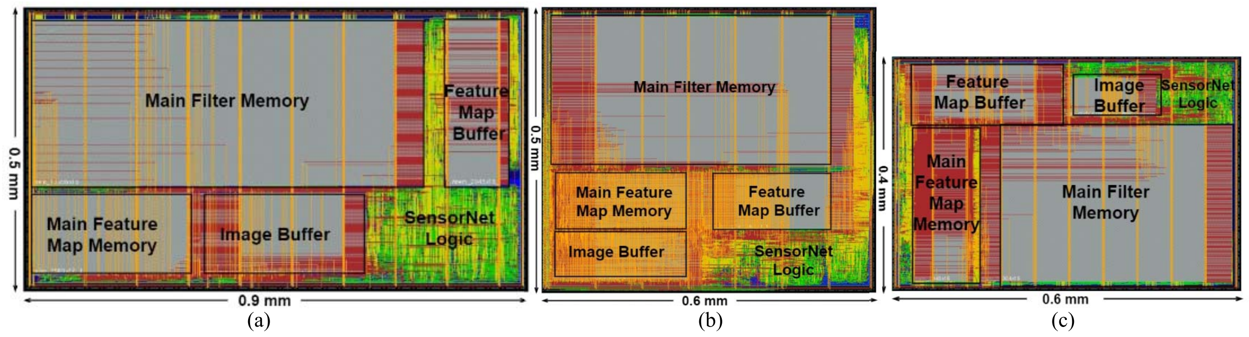

Fig. 19. Post-layout views of SensorNet ASIC implementation in 65 nm, TSMC CMOS technology with operating frequency of 100 MHz. (a) Physical activity monitoring. (b) sdTDS. (c) Stress detection.

TABLE IV
REAL-TIME SENSORNET IMPLEMENTATION RESULTS ON NVIDIA JETSON TX2 SoC (CPU+GPU)

Merits/Case Studies	Activity Monitoring		sdTDS		Stress Detection	
	CPU	GPU	CPU	GPU	CPU	GPU
Latency (ms)	3.1	0.9	2.1	0.8	1.1	0.75
Throughput (label/s)	316	1185	476	1277	884	1345
Total Power (mW)	1639	1763	1617	1740	1608	1722
Total Energy (mJ)	5.2	1.5	3.4	1.4	1.8	1.3

TABLE V
SensorNet IMPLEMENTATION RESULTS ON CPU, GPU, FPGA AND ASIC IN TERMS OF LATENCY, THROUGHPUT, POWER AND ENERGY CONSUMPTION, FOR THE PHYSICAL ACTIVITY MONITORING CASE STUDY

Metrics/Platform	TX2 CPU	TX2 GPU	FPGA Serial	FPGA Parallel	ASIC 65-nm
Clock frequency (MHz)	350	140	100	100	100
Latency (ms)	3.1	0.9	14.8	2	14.8
Throughput (label/s)	316	1185	67	491	67
Power (mW)	1639	1763	116	175	18.5
Energy (mJ)	5.2	1.5	1.7	0.35	0.27

C. Off-the-Shelf Platforms Implementation Results and Analysis

The proposed SensorNet is built and implemented on the NVIDIA Jetson TX2 platform using the TensorFlow framework. Table IV shows the real-time implementation results in terms of latency, throughput, power and energy consumption for all three case studies. The TX2-CPU refers to using a single CPU core running at the lowest clock frequency of 345 MHz, without using the GPU. As can be seen from the table, on the base setting the energy consumption of the proposed processor is 3.46 mJ, on average (average over the three case studies). Also, it takes 2.1 ms on average to finish all the necessary computations. For the same input data, when the GPU is enabled and processing is performed on the GPU with a clock frequency of 140 MHz and one CPU core is on, running at 345 MHz, on average the energy consumption is 1.4 mJ and the execution time is 0.8 ms which shows an improvement of 2.5× and 2.6× compared to the TX2-CPU setting, respectively.

D. Comparison of SensorNet Performance on Different Embedded Platforms

Table V shows a comparison between our implementation on different embedded platforms including FPGA, ASIC and NVIDIA TX2 SoC. The TX2 with GPU is configured when the processing is performed on the GPU with a clock frequency

TABLE VI

SensorNet Detection Accuracy Results Comparison With Other Work for Physical Activity Monitoring, sdTDS and Stress Detection Case Studies

Merits/Case Studies	Physical Activity Monitoring				Tongue Drive System		Stress Detection	
	[8]	[14]	[17]	**This work**	[22]	**This work**	[23]	**This work**
Technique	Multi-channel DCNN	Deep LSTM	PCA+NN	SensorNet	EMI+LR	SensorNet	Gaussian Mixture Model	SensorNet
Accuracy (%)	93	85	89	98	96	96	85	94

of 140 MHz and a single CPU core is on, running at 345 MHz. For the FPGA-based SensorNet we include both fully-serial and fully-parallel architecture when implemented on Artix-7 platform. As it can be seen from the table, ASIC implementation achieves lowest power and energy consumption compared to other platforms. Also, power and energy consumption of our fully-parallel FPGA implementations is 9× and 15× lower compared to CPU implementation. Also, compared to GPU implementation, it achieves 10× and 4× lower power and energy consumption.

E. Comparison With Existing Work

Recently several FPGA/ASIC accelerators have been proposed for DCNN, mainly for computer vision applications. Andri *et al.* [29] proposed an architecture for Ultra-Low power Binary-Weight CNN acceleration. They proposed to use latch-based SCMs for on-chip data storage to be able to scale down the operating voltage even further. They implemented the accelerator using UMC 65-nm technology which consumes 153 mW. Also, Wang *et al.* [30] proposed an energy-efficient architecture for Binarized DCNN. Authors employed different architectural level and circuit level optimization techniques to reduce off-chip I/O access and power consumption. They implemented their architecture using ASIC 130-nm CMOS technology which consumes 842 mW power for VGG-16 network. Chen *et al.* [31] proposed a processing dataflow, called row stationary to minimize data movement from on-chip and off-chip memories. They implemented the optimizaed accelerator using ASIC 65-nm CMOS technology which consumes 236 mW for VGG-16 network. Alemdar *et al.* [28] proposed to train Ternary Neural Networks using a teacher-student approach based on a novel, layer-wise greedy methodology. They implemented their proposed work on FPGA and ASIC which consumes 6.25 W and 0.377 W, respectively. Zhao *et al.* [32] proposed an FPGA-based accelerator for Binarized DCNN which uses variable-width line buffer. They implemented the accelerator on Zynq SoC device which consumes 4.7 W power. Motamedi *et al.* [33] proposed a DCNN accelerator and explore available sources of parallelism to minimize the execution. They evaluated their proposed solution using AlexNet on Vertix-7 FPGA. Nakahara and Sasao [34] proposed a new MAC architecture for DCNN. The proposed MAC is decomposed into 4-bit parallel units. They evaluated their proposed solution using ImageNet on Virtex7 FPGA. Li *et al.* [35] proposed a structural design optimization for Deep Convolutional Neural Networks using Stochastic Computing. They evaluated their designs using LeNet-5 and MNIST datasets and they implemented their proposed technique on CPU, GPU and Binary ASIC. Basterretxea *et al.* [17] proposed a solution for multimodal activity recognition on FPGA with 268 mW power consumption.

Compared to the previous work which mainly target FPGA/ASIC hardware acceleration for the computer vision applications, this paper proposes a customized DCNN architecture for multimodal time-series data classification. We employed algorithmic optimization techniques to reduce the memory requirements and number of operations while achieving high detection accuracy. Also, we proposed a hardware architecture which is scalable and allows for different parallel and fixed-point representations. We employed different hardware optimization techniques to reduce memory access, power consumption and resource utilization. Therefore, customizing and optimizing DCNN for multimodal time-series applications, along with an efficient hardware architecture design result in low power and energy consumption in this work.

Table VI shows SensorNet detection accuracy results comparison with other work for Physical Activity Monitoring, sdTDS and Stress Detection case studies. Based on the results, SensorNet achieves higher or comparable detection accuracy for all three different case studies. For Activity Recognition case study, SensorNet achieves 5%, 13% and 9% higher detection accuracy compared to [8], [14], and [17]. Also, for Tongue Drive system the detection accuracy of SensorNet is similar to [22]. Furthermore, SensorNet achieves 9% higher accuracy compared to [23], for Stress detection case study.

Table VII shows a comparison of the proposed SensorNet hardware implementation results with existing deep learning solutions on embedded devices. When SensorNet is deployed at Xilinx FPGA device with a fully-parallel design and running at 100 MHz, it consumes 154 mW and 175 mW power for Stress Detection and Activity Recognition case studies, respectively. For the same Activity Recognition case study, FPGA-based SensorNet consumes 1.7×, 73× and 123× lower power, latency and energy compared to [17]. SensorNet with a fully-parallel architecture for Activity Monitoring case study, uses 10 DSP Slices, 35 BRAM, 2500 Slices. Whereas [17], uses 19 DSP Slices, 65 BRAM, 2175 Slices. Therefore, SensorNet utilize less DSP slices and BRAM compared to [17], but slightly more number of slice.

When SensorNet is implemented using ASIC 65-nm and running at approximately 900 MHz, it consumes 60 mW and 102 mW power for Stress Detection and Activity Recognition case studies, respectively. Since we could not find any ASIC work for a similar dataset, we compare SensorNet results for Stress Detection case study with [28] results for CIFAR-10, which is a small computer vision dataset. Compared to [28], SensorNet consumes 6× and 2.2× lower power and energy but 2.6× higher latency. This paper focuses on multimodal

TABLE VII
COMPARISON OF SensorNet PERFORMANCE WITH RELATED WORKS WHEN EVALUATED ON REAL-TIME EMBEDDED SETTINGS

Metrics	[13]	[18]	[17]	[28]	**This work**		**This work**	
Application	Human Activity Recognition			CIFAR-10	Stress Detection		Human Activity Recognition	
No. of channels	6	6	40	–	7		40	
Technique	DeepSense	Multimodal RBM	PCA+NN	Ternary NN	SensorNet		SensorNet	
Platform	Intel Edison	Snapdragon	FPGA	ASIC 28-nm	FPGA	ASIC 65-nm	FPGA	ASIC 65-nm
Latency (ms)	105	50	146	0.3	1	0.8	2	1.6
Power (W)	6	1.9	0.294	0.377	0.154	0.06	0.175	0.102
Energy (mJ)	700	96	43	0.111	0.15	0.05	0.35	0.164

time-series case studies which usually do not need to run very fast; therefore, SensorNet on ASIC is not designed to achieve a very low latency compared to previous DCNN on ASIC for computer vision applications.

Compared to [13] and [18], FPGA-based SensorNet achieves $38\times$, $12\times$ lower power, $105\times$, $50\times$ lower latency and $4666\times$, $640\times$ lower energy, for the applications with similar number of data channels.

VIII. CONCLUSION

This paper presents Sensornet, a low power embedded deep convolutional neural network for multimodal time series signal classification. First, the time series data generated by different sensors are converted to images (2-D signals) and then a single DCNN is employed to learn features over multiple modalities and perform the classification task. The proposed SensorNet is scalable as it can process different types of time series data with variety of input channels and sampling rates. Also, it does not need to employ separate signal processing techniques for processing the data generated by each sensor modality. Furthermore, there is no need of expert knowledge for extracting features for each sensor data. Moreover, SensorNet has a very efficient architecture which makes it suitable to be deployed at IoT and wearable devices. The proposed solution was tested using three real-world case studies including Physical Activity Monitoring, dual-mode Tongue Drive system and Stress detection and based on the results, it achieved an average classification accuracy of 98%, 96.2%, 94% for each application, respectively. A custom low power hardware architecture is also designed for the efficient deployment of SensorNet at embedded real-time systems and when is implemented on Xilinx FPGA (Artix-7), on average consumes 246 μJ energy. To further reduce the power consumption, SensorNet is implemented using ASIC at the post layout level in 65-nm CMOS technology which consumes approximately $8\times$ lower power compared to the FPGA implementation. Additionally, SensorNet is implemented on NVIDIA Jetson TX2 SoC (CPU+GPU) and compared to TX2 single-core CPU and GPU implementations, FPGA-based SensorNet obtains $15\times$ and $4\times$ improvement in energy consumption.

REFERENCES

[1] A. Kampouraki, G. Manis, and C. Nikou, "Heartbeat time series classification with support vector machines," *IEEE Trans. Inf. Technol. Biomed.*, vol. 13, no. 4, pp. 512–518, Jul. 2009.

[2] A. Jafari, A. Page, C. Sagedy, E. Smith, and T. Mohsenin, "A low power seizure detection processor based on direct use of compressively-sensed data and employing a deterministic random matrix," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2015, pp. 1–4.

[3] A. Reiss and D. Stricker, "Introducing a modular activity monitoring system," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug./Sep. 2011, pp. 5621–5624.

[4] S. Seto, W. Zhang, and Y. Zhou, "Multivariate time series classification using dynamic time warping template selection for human activity recognition," in *Proc. IEEE Symp. Ser. Comput. Intell.*, Dec. 2015, pp. 1399–1406.

[5] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Very Large Data Base Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[6] Y. Chen *et al.*, "The UCR time series classification archive," Dept. Comput. Sci. Eng., Univ. California, Riverside, Riverside, CA, USA, Tech. Rep., Jul. 2015. [Online]. Available: www.cs.ucr.edu/~eamonn/time_series_data/

[7] A. Jafari, M. Hosseini, A. Kulkarni, C. Patel, and T. Mohsenin, "BiNMAC: Binarized neural network manycore accelerator," in *Proc. 28th Great Lakes Symp. VLSI*, 2018, pp. 443–446.

[8] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Proc. Int. Conf. Web-Age Inf. Manage.* Cham, Switzerland: Springer, 2014, pp. 298–310.

[9] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Proc. 29th Workshops AAAI Conf. Artif. Intell.*, 2015, pp. 40–46.

[10] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[11] P. Vepakomma, D. De, S. K. Das, and S. Bhansali, "A-Wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities," in *Proc. IEEE 12th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, Jun. 2015, pp. 1–6.

[12] X. Li *et al.* (2017). "Concurrent activity recognition with multimodal CNN-LSTM structure." [Online]. Available: https://arxiv.org/abs/1702.01638

[13] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 351–360.

[14] Y. Guan and T. Ploetz. (2017). "Ensembles of deep LSTM learners for activity recognition using wearables." [Online]. Available: https://arxiv.org/abs/1703.09370

[15] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 1307–1310.

[16] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng. (2017). "Cardiologist-level arrhythmia detection with convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1707.01836

[17] K. Basterretxea, J. Echanobe, and I. del Campo, "A wearable human activity recognition system on a chip," in *Proc. Conf. Design Archit. Signal Image Process. (DASIP)*, Oct. 2014, pp. 1–8.

[18] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. Adjunct*, 2016, pp. 185–188.

[19] M. Li *et al.*, "Multimodal physical activity recognition by fusing temporal and cepstral information," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 4, pp. 369–380, Aug. 2010.

[20] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6a overview of mini–batch gradient descent," Coursera, Mountain View, CA, USA, Tech. Rep., 2012. [Online]. Available: https://class.coursera.org/neuralnets-2012-001/lecture/

[21] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Proc. 16th Int. Symp. Wearable Comput. (ISWC)*, Jun. 2012, pp. 108–109.

[22] A. Jafari, N. Buswell, M. Ghovanloo, and T. Mohsenin, "A low-power wearable stand-alone tongue drive system for people with severe disabilities," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 58–67, Feb. 2018.

[23] J. Birjandtalab, D. Cogan, M. B. Pouyan, and M. Nourani, "A non-EEG biosignals dataset for assessment and visualization of neurological status," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2016, pp. 110–114.

[24] F. Chollet *et al.*, "Keras," Tech. Rep., 2015. [Online]. Available: https://keras.io

[25] A. Jafari, M. Ghovanloo, and T. Mohsenin, "An embedded FPGA accelerator for a stand-alone dual-mode assistive device," in *Proc. IEEE Biomed. Circuits Syst. Conf.*, vol. 101, Oct. 2017, pp. 1–4.

[26] A. Page, A. Jafari, C. She, and T. Mohsenin, "Sparcnet: A hardware accelerator for efficient deployment of sparse convolutional networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, 2017, Art. no. 31.

[27] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Programm. Gate Arrays (FPGA)*, New York, NY, USA, 2015, pp. 161–170.

[28] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Pétrot, "Ternary neural networks for resource-efficient AI applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2547–2554.

[29] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An architecture for ultralow power binary-weight CNN acceleration," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 48–60, Jan. 2018.

[30] Y. Wang, J. Lin, and Z. Wang, "An energy-efficient architecture for binary weight convolutional neural networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 2, pp. 280–293, Feb. 2018.

[31] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[32] R. Zhao *et al.*, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2017, pp. 15–24.

[33] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, "Design space exploration of FPGA-based deep convolutional neural networks," in *Proc. 21st Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2016, pp. 575–580.

[34] H. Nakahara and T. Sasao, "A deep convolutional neural network based on nested residue number system," in *Proc. 25th Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2015, pp. 1–6.

[35] Z. Li *et al.*, "Structural design optimization for deep convolutional neural networks using stochastic computing," in *Proc. Design Automat. Test Eur. Conf. Exhib.*, Mar. 2017, pp. 250–253.

Ali Jafari received the Ph.D. degree in computer science and electrical engineering from the University of Maryland Baltimore County in 2017. His current research interests include low-power digital application-specified integrated circuit and field-programmable gate array designs, efficient hardware architecture design for deep learning accelerators, hardware-aware deep neural networks and machine learning algorithms, and digital signal processing for low-power Internet of Things and wearable devices.

Ashwinkumar Ganesan received the B.S. degree in engineering from the Maharashtra Institute of Technology, India. He is currently pursuing the Ph.D. degree with the Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD, USA. His primary areas of research are deep neural networks with focus on vision and language to design neural dialog systems. He has a wide range of research interests from modeling time series for low-power devices to large-scale machine learning.

Chetan Sai Kumar Thalisetty is currently pursuing the M.S. degree with the Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, MD, USA. His research interests mainly focus on software–hardware embedded systems, deep neural networks, and machine learning.

Varun Sivasubramanian is currently pursuing the M.S. degree with the Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, MD, USA. His research interests mainly focus on low-power application-specified integrated circuit designs, embedded systems, and machine learning.

Tim Oates received the B.S. degree in computer science and electrical engineering from North Carolina State University in 1989 and the M.S. and Ph.D. degrees from the University of Massachusetts Amherst in 1997 and 2000, respectively. He held a post-doctoral position with the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, for one year. He is currently a Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County. He has authored or co-authored over 150 peer reviewed papers. His research interests include pattern discovery in time series, grammatical inference, graph mining, statistical natural language processing, robotics, and language acquisition. He is a member of the Association for Computing Machinery and the Association for the Advancement of Artificial Intelligence. He was a recipient of the prestigious NSF CAREER Award in 2004.

Tinoosh Mohsenin received the M.S. degree in electrical and computer engineering from Rice University in 2004 and the Ph.D. degree in electrical and computer engineering from the University of California, Davis, in 2010. She is currently an Associate Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, where she is also the Director of the Energy Efficient High Performance Computing Lab. She has authored or co-authored over 80 peer-reviewed journal and conference publications. Her research focus is on designing highly accurate and energy efficient embedded processors for machine learning, signal processing and knowledge extraction techniques for autonomous systems, wearable smart health monitoring, and embedded big data computing. She served as a Technical Program Committee Member for the IEEE International Solid-State Circuits Conference Student Research (ISSCC-SRP), the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS (TBioCAS), the IEEE International Symposium on Circuits and Systems, the ACM Great Lakes Symposium on VLSI, and the IEEE International Symposium on Quality Electronic Design conferences. She was a recipient of the NSF CAREER Award in 2017. She received the Best Paper Award at the GLSVLSI Conference in 2016 and the Best Paper Honorable Award at ISCAS 2017 for developing domain-specific accelerators for biomedical, deep learning, and cognitive computing. She served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I and the IEEE TBioCAS. She was the Local Arrangement Co-Chair of the 50th IEEE International Symposium on Circuits and Systems in Baltimore. She also serves as a Secretary for the IEEE P1890 on Error Correction Coding for Non-Volatile Memories.