APPROVAL SHEET

Title of Dissertation: A Scalable and Low Power Deep Convolutional Neural Network for Multimodal Data Classification In Embedded Real-Time Systems

Name of Candidate: Ali Jafari
Doctor of Philosophy, 2017

Dissertation and Abstract Approved: Tinoosh Mohsenin
(Tinoosh Mohsenin)
(Assistant Professor)
(Department of Computer Science
and Electrical Engineering)

Date Approved: 11/27/2017

ABSTRACT

Title of dissertation: A Scalable and Low Power Deep Convolutional Neural Network for Multimodal Data Classification in Embedded Real-Time Systems

Name of Candidate: Ali Jafari
Computer Engineering, 2017

Dissertation directed by: Professor Tinoosh Mohsenin
Department of Computer Science & Electrical Engineering

Multimodal time series signals are generated by different sensors such as accelerometers, magnetometers, gyroscopes and heart rate monitors, where each sensor usually has various number of input channels and sampling rates. Different signal processing techniques such as feature extraction and classification are employed to process the data generated by each sensor modality which: 1) can lead to a long design time, 2) requires expert knowledge in designing the features, and 3) is unscalable when adding new sensors. Moreover, with recent advances in Internet of Things (IoT) and wearable devices, a major challenge is the ability to efficiently deploy the multimodal signal processing techniques in embedded, resource-bound settings that have strict power and area budgets.

In this dissertation we target the previously mentioned challenges. In the first contribution, we propose "SensorNet" which is a scalable deep convolutional neural network designed to classify multimodal time series signals. The raw time series signals generated by different

sensor modalities with different sampling rates are first fused into images; then, a Deep Convolutional Neural Network (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. (6) has a very efficient architecture which makes it suitable to be employed at IoTs and wearable devices.

In the second contribution, we propose a custom low power hardware architecture for the efficient deployment of SensorNet at resource-limited embedded devices, which can perform the entire SensorNet signal processing in real-time with minimal energy consumption. The proposed architecture is fully reconfigurable for different applications with various requirements.

Finally, we propose a stand-alone dual-mode Tongue Drive System (sdTDS) which employs SensorNet to perform all required multimodal signal processing in real-time. sdTDS 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 and head movements.

SensorNet performance is evaluated using three different case studies including Physical

Activity Monitoring, sdTDS and Stress Detection and it achieves an average detection accuracy of 98%, 96.2% and 94% for each case study, respectively. Furthermore, we implement SensorNet using our custom hardware architecture on Xilinx FPGA (Artix-7) which consumes 17 mJ, 9 mJ and 3.5 mJ energy for Physical Activity Monitoring, sdTDS and Stress Detection case studies, respectively. To further reduce the power consumption, SensorNet is implemented using ASIC at the post layout level in 65-nm CMOS technology which consumes approximately 7$\times$ lower power compared to the FPGA implementation. Additionally, SensorNet is implemented on NVIDIA Jetson TX2 SoC (CPU+GPU) which is an embedded commercial off-the-shelf platform. Compared to TX2 single-core CPU and GPU implementations, FPGA-based SensorNet obtains 8$\times$ and 12$\times$ improvement in power consumption, and 71$\times$ and 3$\times$ improvement in energy consumption. Furthermore, SensorNet achieves 200$\times$, 63$\times$, 27$\times$ lower energy consumption compared to previous related work.

SensorNet is considered as a generic deep neural network that can accommodates a wide range of applications with minimal effort.

A Scalable and Low Power Deep Convolutional Neural Network for Multimodal Data Classification in Embedded Real-Time Systems

by

Ali Jafari

PhD Dissertation, 2017

Advisory Committee:
Professor Tinoosh Mohsenin, Chair/Advisor
Professor Mohamed Younis
Professor Tim Oates
Professor Fow-Sen Choa
Professor Maysam Ghovanloo (Georgia Institute of Technology)

I dedicate this PhD research dissertation to my family, especially to my parents whose encouragement and support have made this journey possible.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Tinoosh Mohsenin. Her enthusiasm and remarkable foresight positively impacted this PhD research. It has been an honor to work with her and learn from her and I always appreciate her countless valuable advice during my PhD. I further would like to thank Professor Maysam Ghovanloo from GT-Bionics lab at Georgia Institute of Technology for lending his vast knowledge in the field of Assistive Technologies. I would like to extend my gratitude to my PhD committee members, namely Dr. Mohamed Younis, Dr. Tim Oates, and Dr. Maysam Ghovanloo and Dr. Fow-Sen Choa for their guidance and very useful feedback.

I would also like to acknowledge lecturers, colleagues, and all of my lab mates from Energy Efficient High Performance Computing (EEHPC) lab. Finally, I especially thank my family for their love, encouragement and support throughout the course of this research. Words cannot express how grateful I am to my parents for their sacrifices.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation and Problem Statement

Time series data is a generalized form of data that is gathered in different kinds of domains from healthcare [Kampouraki, Manis, & Nikou] 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 [Reiss & Stricker2011], to engines in cars and power plants [Yan & Yu2015] using sensors. Modeling and classifying time series thus has a wide range of applications. All these datasets are represented by a time series which is univariate or multivariate depending on the number of sensor modalities being measured. Multivariate (Multimodal) signals are generated by different sensors usually with different sampling frequencies such as accelerometers, magnetometers, gyroscopes and heart rate monitors.

Traditionally, time series classification problems have been solved with approaches like Dynamic Time Warping (DTW) [Batista, Wang, & Keogh2011, Seto, Zhang, & Zhou2015] and k-nearest neighbor (k-NN) [Ding *et al.*2008]. These methods or a combination of them provide a benchmark for current time series classification research [Chen *et al.*2015]. Different signal processing techniques such as feature extraction and

classification are employed to process the data generated by each sensor modality which: 1) can lead to a long design time, 2) requires expert knowledge in designing the features, 3) requires new algorithm development and implementation if new sensors are employed, which is tedious, 4) needs extensive signal pre-processing, and 5) is unscalable for different real-time applications.

Deep Convolutional Neural Networks (DCNN) have become extremely popular over the last few years after their success during the Imagenet challenge [Krizhevsky, Sutskever, & Hinton2012]. Supervised CNNs are used to perform a large number of tasks such as object detection [Krizhevsky, Sutskever, & Hinton2012], image segmentation [Girshick *et al.*2014], and are combined with Recurrent Neural Networks (RNN) to generate captions for images [Xu *et al.*2015] as well as to recognize speech [Graves, Mohamed, & Hinton2013]. Inspired by these developments, deep networks are applied to classify time series data, perform event detection and engineer features from the input data [Zheng *et al.*2014, Wang & Oates2015, Ordóñez & Roggen2016, Vepakomma *et al.*2015, Li *et al.*2017, Yao *et al.*2017, Guan & Ploetz2017, Jiang & Yin2015, Rajpurkar *et al.*2017, Yan & Yu2015]. However, these solutions encounter various challenges such as: Low detection accuracy, high latency, large and power-hungry architectures when deployed at Internet of Things (IoT) and wearable devices.

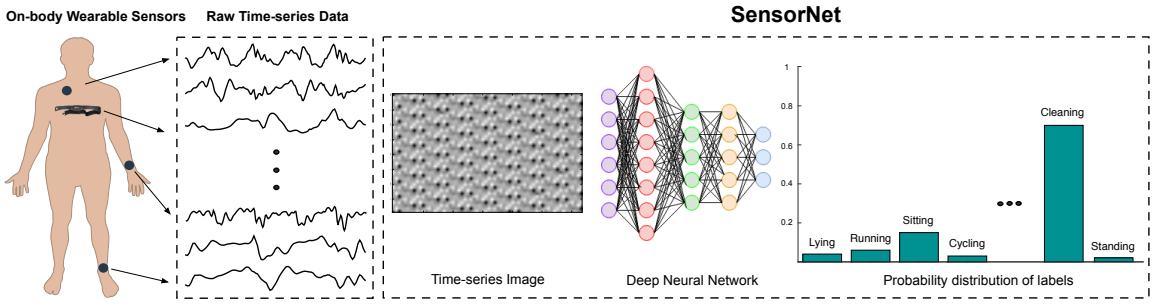

FIG. 1.1. SensorNet high-level diagram

1.2 Contributions

In this dissertation, SensorNet shown in Figure 1.1 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. The time-series signals generated by different sensor modalities with different sampling rates are first fused into images and then a Deep Convolutional Neural Network is utilized to model the time series data while it learns to correlate the information in the signals from different sensor modalities, simultaneously. 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) 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. (6) has a very efficient architecture which makes it suitable to be employed in low

power and resource-bound embedded devices.

Furthermore, we propose a custom hardware architecture for an efficient deployment of SensorNet on low power and resource-limited embedded devices, which can perform the entire SensorNet signal processing in real-time.

Also, we propose a stand-alone dual-mode Tongue Drive System (sdTDS) which uses SensorNet to perform all required signal processing in real-time. sdTDS 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 and head movements.

We evaluate the SensorNet performance for three different case studies including Physical Activity Monitoring, sdTDS and Stress Detection on different embedded settings such as FPGAs, ASICs and NVIDIA Jetson TX2 SoC (CPU+GPU). To summarize, the dissertation provides the following key contributions:

- Propose SensorNet which is scalable Deep Convolutional Neural Networ to classify multimodal time series signals
- Perform extensive hyperparameter optimization for SensorNet with the goal of reducing memory requirements, hardware complexity and power consumption while achieving high detection accuracy
- Propose a custom hardware architecture for an efficient deployment of SensorNet on low power and resource-limited embedded devices, which can perform the entire

SensorNet signal processing in real-time

- Develop and build sdTDS, a wireless assistive technology which employes SensorNet for it's multi-modal signal classification. Individuals with severe disabilities can use sdTDS to potentially control their environment using their voluntary tongue and head movements

- Evaluate SensorNet in terms of detection accurcay, memory requirements and number of computations using three real-world case studies including Physical Activity Monitoring, dual-mode Tongue Drive System and Stress Detection

- Implement SensorNet on low power FPGAs and also ASIC through full place and route flow in 65-nm CMOS technology and provide results and analysis in terms of power consumption, latency and resource utilization

- Implement SensorNet on NVIDIA Jetson SoC TX2 commercial platform and provides comparisons with FPGAs and ASIC performance results

1.3 Organization of Dissertation

The dissertation consists of the following chapters as an approach to the contributions and objectives stated above.

- Chapter 2 introduces state-of-the-art for multimodal data classifications using different deep neural networks. Also, it provides related prior work for hardware implementations

of deep neural networks on embedded platforms.

- Chapter 3 describes the proposed SensorNet architecture and discuss all the required neural network layers.
- Chapter 4 provides SensorNet experimental results when it is applied to three different real-world case studies. Also, different deep neural network hyperparameter optimizations are explored in this section.
- Chapter 5 discusses the proposed hardware architecture for SensorNet in details. Furthermore, employed hardware optimization techniques are discussed in this chapter.
- Chapter 6 provides SensorNet hardware implementation results on FPGA and ASIC platforms targeting three case studies and compared against different off-shelf platforms.
- Chapter 7 finally concludes and provides paths for future work to extend upon this dissertation.

Chapter 2

BACKGROUND

2.1 Overview

In recent years, several multimodal data classification approaches have been proposed which are discussed in this section. These approaches have been proposed for different applications such as Human Activity Recognition, Congestive Heart Failure Detection, Mobile Sensing Data Processing. Furthermore, a number of solutions on deep convolutional neural network for low power embedded settings have been proposed which will be discussed in this section.

2.2 Related Prior Work

2.2.1 Multimodal Data Classification

[Wang & Oates2015] uses standard CNN architectures like Tiled Convolutional Neural Networks by first converting a time series into an image before using a CNN to classify it. The authors convert 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 model each variable separately before correlating them or require preprocessing the stream into an image. In [Zheng *et al.*2014] authors 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 & Roggen2016] 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. They showed they achieved better accurcay results compared to baseline CNN mdoel. [Vepakomma *et al.*2015] 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 with average test accuracy of 90%. Their network consists two hidden layers. Their proposed network is designed specifically for their sensing system. Authors in [Li *et al.*2017] 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. In [Yao *et al.*2017] authors 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 on Nexus5 and Intel Edison and reported latency and power consumption results. [Guan & Ploetz2017] 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 & Yin2015] proposed an approach for the activity recognition task that, first assembles signal sequences of accelerometers and gyroscopes into a novel activity image. Then 2D Discrete Fourier Transform is applied to the signal image and its magnitude is chosen as their activity image and as input to the DCNN. [Rajpurkar *et al.*2017] proposed a Cardiologist-Level Arrhythmia Detectiona using a 34-layer CNN and they exceed the average cardiologist performance in both sensitivity and precision.

2.2.2 Deep Neural Networks

In recent years, Deep Neural Networks (DNNs) have become extremely popular because of their outstanding results in the areas such as computer vision [Krizhevsky, Sutskever, & Hinton2012], voice recognition [Hinton & others2012], natural language processing [Collobert & Weston2008], robotics [Hwu & others2017] and time series data classification [Zheng *et al.*2014]. However, due to intensive computations and large memory requirements, implementing deep neural networks on resource limited and low

power embedded platforms is challenging, specially when dealing with a network with many fully connected or convolution layers [Page & others2017]. Also, performance becomes limited by memory bandwidth to access the training weights saved usually in off-the-chip memory. Several methods have been proposed to address efficient training and inference in deep neural networks [Rastegari & others2016]: Shallow networks [Dauphin & Bengio2013], quantizing parameters [Gong & others2014], network binarization [Courbariaux & others2015] and compressing pre-trained deep networks [Han, Mao, & Dally2015]. These methods improve the efficiency of DNNs for implementing on low-power embedded platforms, however, even by employing these methods, an off-the-chip memory is usually required to save the model weights. Moreover, power consumption is still high due to the need of accessing off-the-chip memory constantly. For wearable devices and Internet of Things (IoT) platforms, the power consumption, size and real-time requirements are even more restricted. In 2016, [Courbariaux *et al.*2016] proposed Binarized Neural Networks (BNNs) to address the previously mentioned challenges. BNN has a great compact representation of network weights and activation values compared to a standard DNN by constraining each value to either +1 or -1 (binary). During the forward pass BNNs: 1) Drastically reduce the memory requirements since the model weights can be stored in one single bit (-1 can be stored as 0 and +1 stored as 1). 2) Eliminate the need of using off-the-chip memory in some popular networks. 3) Replace multiply operations with bit-wise operations (mostly XNOR) which improves power efficiency. Since 2016, different works [Courbariaux & others2015, Rastegari &

others2016] have shown that BNNs can achieve comparable accuracies to full-precision DNNs for some popular datasets such as MNIST, CIFAR10 and ImageNet. [Zhao & others2017, Umuroglu & others2017, Nurvitadhi & others2016] have proposed BNN hardware accelerators on CPU, GPU, FPGA and ASIC. Although BNNs are very efficient to be employed in low power and resource-limited embedded devices, they have not shown high detection accuracy for multimodal time series data classifications yet.

In summary, most of the previous work do not propose a real-time hardware solution or general-purpose processors have been used [Radu *et al.*2016, Yao *et al.*2017] which results in high power consumption. Also, the architecture developed in previous work are not efficient for hardware deployment at IoT and wearable devices [Zheng *et al.*2014, Guan & Ploetz2017]. Furthermore, some of the previous related work require expert knowledge in designing the features which results in a long design time [Li *et al.*2010] and those work are not usually scalable.

Chapter 3

PROPOSED SENSORNET

3.1 Deep Neural Networks Overview

In most of the deep neural networks, there are large variety of layers including Convolutional, Fully-connected, Pooling, Batch normalization layers. Also, there are activation functions such as Sigmoid, Tanh, and ReLU, which can be considered as separate layers. Among the neural network layers, fully-connected and convolutional layers are often the most highly utilized and contain the majority of the complexity in the form of computation and memory requirements. In the following section, we provide a brief explanation about the most commonly used layers including their mathematical formulation and complexity requirements in terms of computation and memory.

3.1.1 Convolutional Layers

Convolutional layers are the core building block of a convolutional neural network. The layers consist of learnable filters banks (sets), which have a small receptive field that

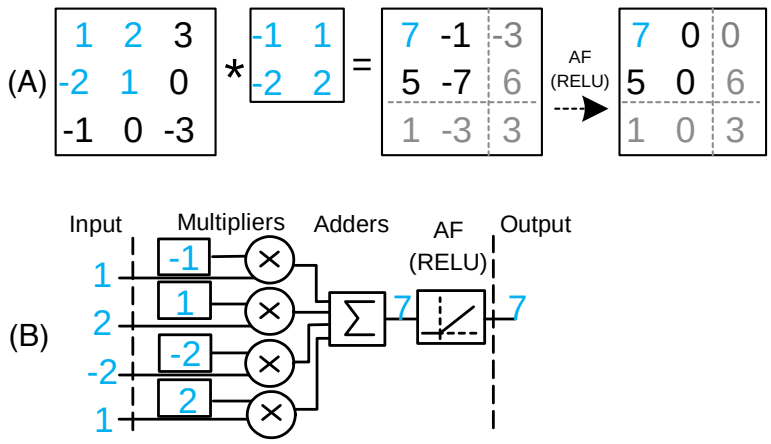

FIG. 3.1. (A) An example of convolving a 3x3 image by a 2x2 filter, (B) A hardware schematic which demonstrates one single convolution operation.

extend through the full depth of the input. During the forward pass, each filter is convolved across the width and height of the input, computing the dot product between the entries of the filter and the input and producing a feature map of that filter. Feature maps for all filters along the depth dimension of the input data, form the full output of the convolution layer. Figure 3.4 shows convolution operation for a 3x3 image by a 2x2 filter followed by an activation function. A hardware schematic which demonstrates one single operation is also depicted. The convolutional layers use a non-linear activation function which will be discussed later.

For a 1-D input $X_{M,C_{in}}$ of length M and with input channels C_{in}, a 1-D convolutional layer with stride S, filter length F, weight $W_{C_{out},C_{in},F}$, feature maps C_{out}, an output signal $Y_{N,C_{out}}$ with length $N = 1+\lfloor (M-F)/S \rfloor$ and output channels C_{out}, the output of a single element of a feature channel is computed by:

$$Y_{i,j} = \sum_{c=1}^{C_{in}} \left(\sum_{f=1}^{F} (X_{f+iS,c} W_{j,c,f}) \right) \; for \; i = 0..N-1, j = 1..C_{out} \tag{3.1}$$

The total amount of memory requirements by the layer corresponds to the number of weights for all of the filters which is $C_{out}C_{in}F$. The total computation required for the layer is $2FC_{in}C_{out}N$.

3.1.2 Pooling Layers

Pooling layers are usually used immediately after convolutional layers and perform dimensionality reduction. These layers also referred to as downsampling layers. What the pooling layers do is simplify the information in the output from the convolutional layer. There are different pooling layers such as max-pooling and average-pooling. Max-pooling reduces the size of the image and and also helps the network to learn abstract features in the signal by maximizing the value across the pooling window. The pooling layers are usually applied independently to each input channel. Given a 1-D input $X_{M,C_{in}}$ of length M and with C_{in} input channels, a 1-D pooling layer with stride S and pooling length P will produce an output signal $Y_{N,C_{in}}$ with length $N = 1 + \lfloor (M-P)/S \rfloor$. This layer does not requires any memory and significantly less computation compared to convolution layers because it is applied independently to each input channel.

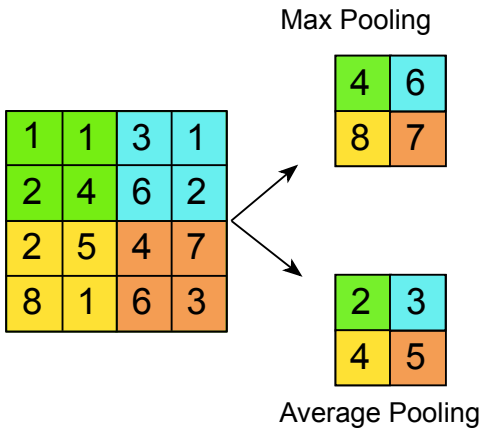

FIG. 3.2. Max-pooling and average-pooling examples with a 2x2 window and stride = 2.

3.1.3 Fully-connected Layers

The fully-connected layer is a traditional Multi Layer Perceptron (MLP) that connects every neuron in the previous layer to every neuron on the next layer. Their activations can thus be computed with a matrix multiplication followed by a bias offset. The main issue with fully-connected layers is that the layer requires significant amount of memory and computation complexity. Given a 1-D input X_M of length M, a fully-connected layer with N neurons, weight $W_{N,M}$ and a 1-D output Y_N with length N, the output for a single neuron is computed by:

$$Y_j = \sum_{m=1}^{M} (X_m W_{j,m}) \ \ for\ j = 1..N \tag{3.2}$$

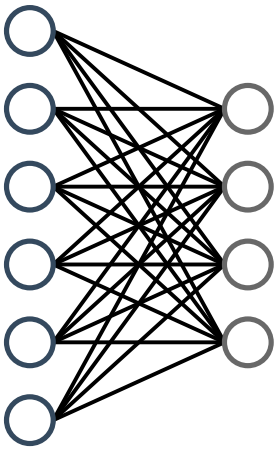

FIG. 3.3. Fully-connected Layer.

The total amount of memory required for the layer corresponds to the total number of weights, NM, and the total computation is approximately $2MN$. Therefore, the memory and computation contribute equally in terms of complexity.

Usually, after several convolutional and max-pooling layers, the high-level reasoning in the neural network is performed through fully-connected layers. Also, a fully-connected layer with Softmax activation function is used in the output layer for the final classification.

3.1.4 Activation Functions

In biologically inspired neural networks, the activation function is usually an abstraction representing the rate of action potential firing of the cell. Activation functions play an important role in the Artificial Neural Network to learn and make sense of Non-linear complex functional mappings between the inputs and response variables and ability to

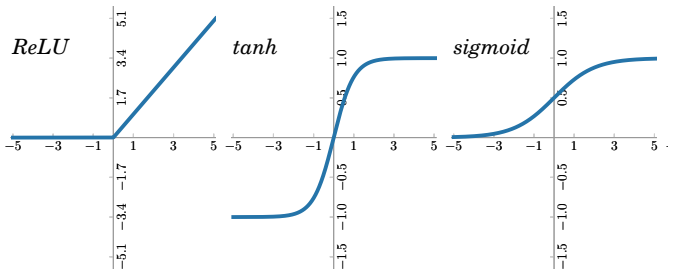

FIG. 3.4. Some common activation functions used in neural networks.

satisfy the profound universal approximation theorem. Figure 3.4 shows some common activation functions used in the Neural Networks including Rectified Linear Unit (ReLU), Hyperbolic tangent (Tanh) and Sigmoid. Convolutional and fully-connected layers use non-linear activation functions. Recently, the most common activation functions are ReLUs which have shown to provide better performance compared to others. A ReLU is represented with the following function:

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

In the ReLUs, the activation is linear when the output is positive and hence does not suffer from a vanishing gradient problem. Also, ReLUs are very efficient for hardware implementation because they require few logics and operations to perform.

3.2 SensorNet Signal Pre-processing

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, thus 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 SensorNet 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 $(t - W + 1)...t$. Thus, the network can look back W prior states of each variable and given the current state of each variable, predicts the label.

3.3 SensorNet Architecture

Fig. 3.5 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

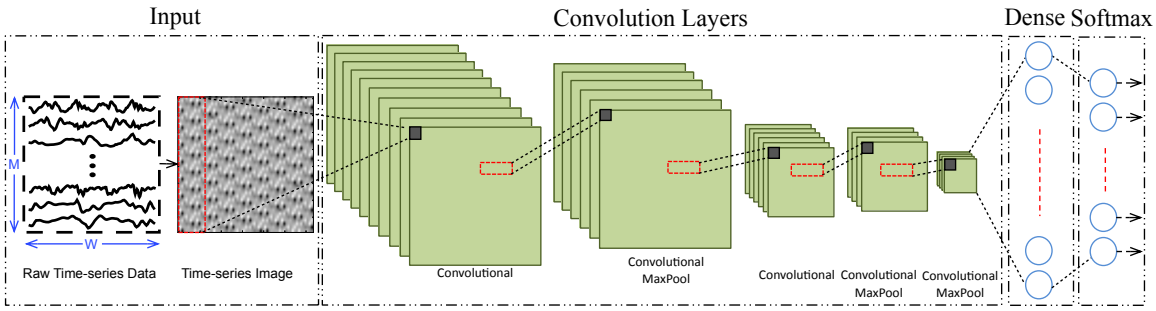

FIG. 3.5. The proposed Sensornet architecture which consists of Convolutional, Fully-connected (Dense) and Softmax layers.

layers. SensorNet architecture, the number of layers and filters and also the filter size for each layer are chosen based on an extensive hyperparameter optimization process. In Chapter 4 Section 4.4 we will discuss the optimization process.

The 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 n or 1 remains unaffected by the ordering of the variables. Therefore, the filter size for the first convolutional laye is $M \times 5$ and 1×5 for other layers, where M is number of input modalities.

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 use a Sigmoid as the activation function 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.

Chapter 4

SENSORNET EVALUATION USING THREE CASE STUDIES

In this chapter, SensorNet is evaluated using three real-world case studies including Physical Activity Monitoring [Reiss & Stricker2012], stand-alone dual-mode Tongue Drive system [Jafari *et al.*2017] and Stress Detection [Birjandtalab *et al.*2016] and in depth analysis and experimental results are provided. First, we provide a description for each case study and then we explore the experimental results in terms of detection accuracy, memory requirements and number of operations. Also, we explore different hyperparameter optimization techniques and discuss how these techniques affect SensorNet performance.

The information for all the case studies are shown in Table 4.1. 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.

Table 4.1. Information for three different case studies including Physical Activity Monitoring, sdTDS and Stress Detection

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

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 [Chollet & others2015] 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 through to the convolutional layers.

4.1 Case Study 1: Physical Activity Monitoring

4.1.1 Dataset Description

Physical Activity Monitoring dataset (PAMAP2) [Reiss & Stricker2012] 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 as shown in Figure 4.1 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

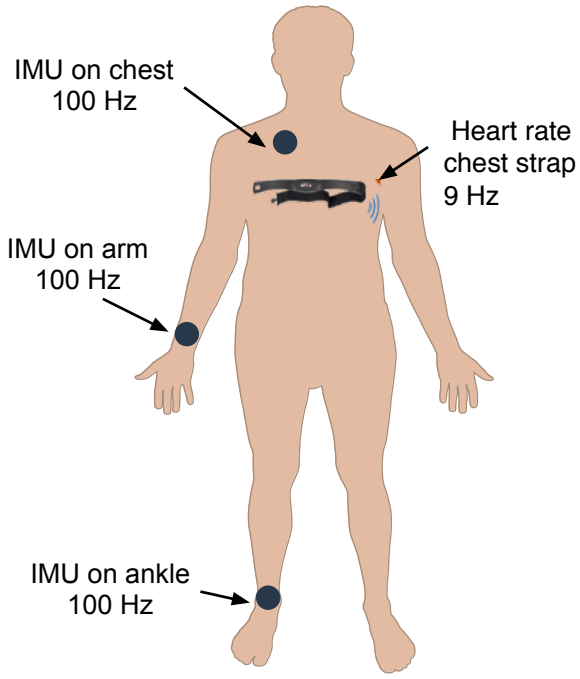

FIG. 4.1. Placement of on-body heart rate monitor and inertial measurements units (IMU) used in the PAMAP2 dataset. Each IMU includes a 3D Accelerometer, Magnetometers, Gyroscope.

of them are valid according to [Reiss & Stricker2012]. Also, out of 9 subjects the data of 8 subjects are used, as subject 9 has a very small number of samples.

Figure. 4.2 shows the class distributions for the PAMAP2 which shows that PAMAP2 is a dataset that was recorded in a constrained manner which results in a well balanced class distribution [Guan & Ploetz2017].

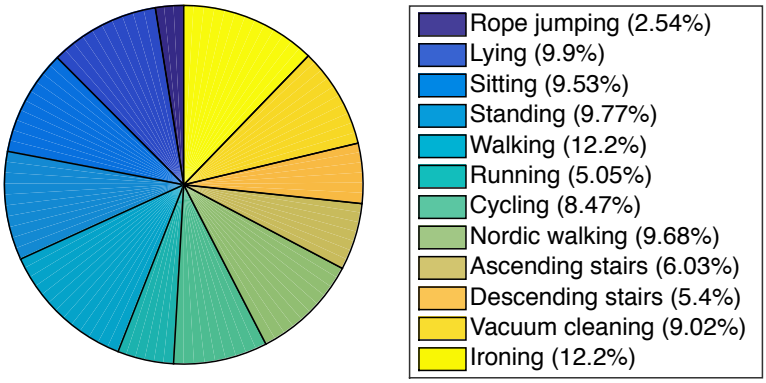

FIG. 4.2. Class distributions of the PAMAP2 dataset used for experimental evaluation.

4.1.2 Data Visualization

In this section we visualize some of the input images, filters and output feature maps. As it was mentioned earlier in this chapter, in the pre-processing stage, SensorNet receives 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. Figure 4.3 show 4 different input images which correspond to four different PAMAP2 activities including Laying, Sitting, Vacuum cleaning and Playing soccer. As it can bee seen from the figure, the images for different activities look totally different. Also, the size of the images are 40×64 because the PAMAP2 dataset has 40 input channels (variables) and the sliding window for this experiment is 64.

Figure 4.4 depicts one set of filters for the second convolutional layer which contains 16 filters. As is shown, each filter has a size of 1×5.

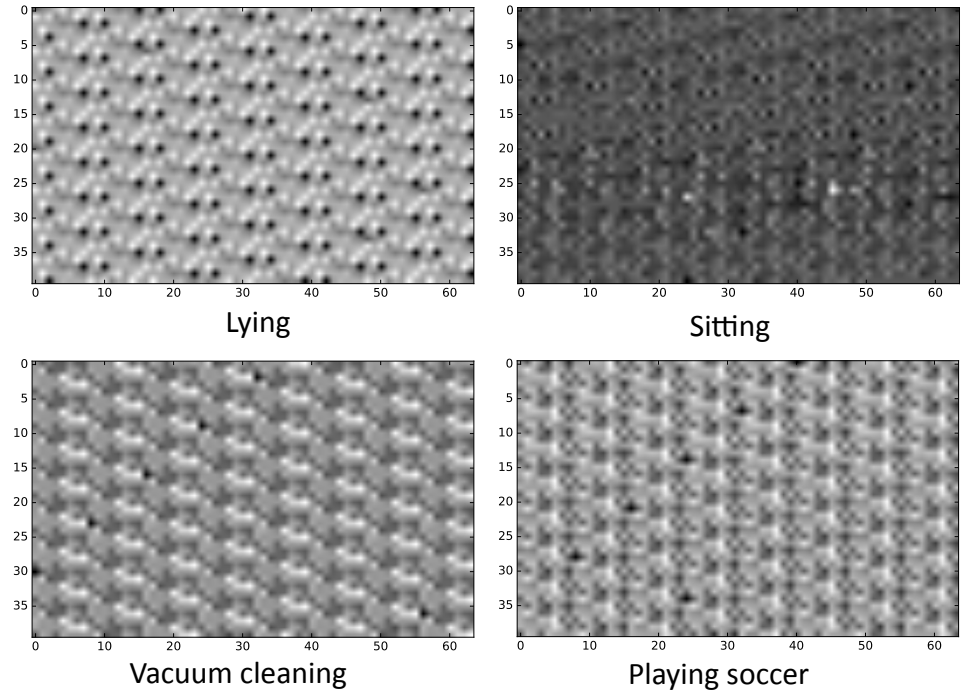

FIG. 4.3. Samples of SensorNet input images for four different activities including Lying, Sitting, Vacuum Cleaning and Playing soccer. Raw input time series data are fuses into these images.

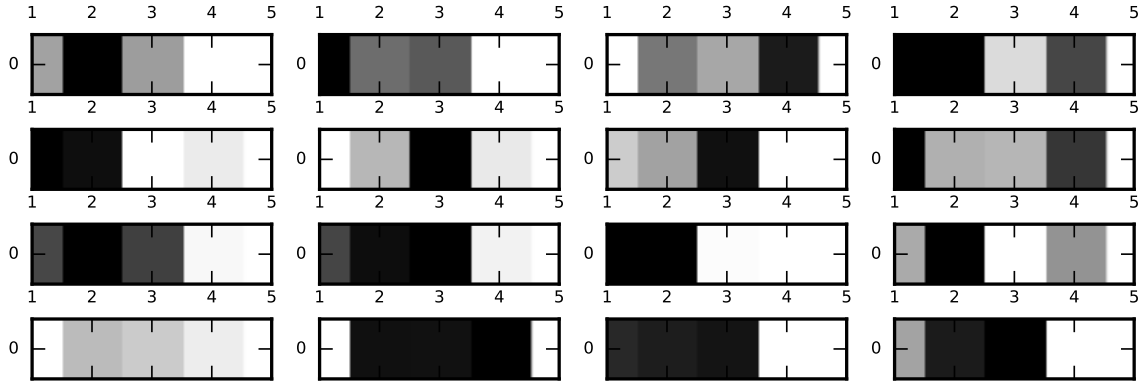

FIG. 4.4. One filter set of SensorNet which contains 16 filters with size of 1×5.

After convolving input images with filters, the output feature maps are generated. Figure 4.5 demonstrates the output feature maps of the convolutional layer after the max-pooling. As it can be seen from the figure, the size of the output feature map is 32 rather than 64 because of the max-pooling layer. Max-pooling layers reduce dimension of the feature maps which reduces the computations and also memory requirements for the fully-connected layer.

4.1.3 Experiment Setup

As we mentioned in Chapter 3 Section 3.3, 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 it can be seen from Figure 4.6, 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.

4.1.4 Experiment Results

After training SensorNet, we evaluate the trained model to determine the detection accuracy. Figure. 4.7 shows the classification accuracy of SensorNet for the Physical

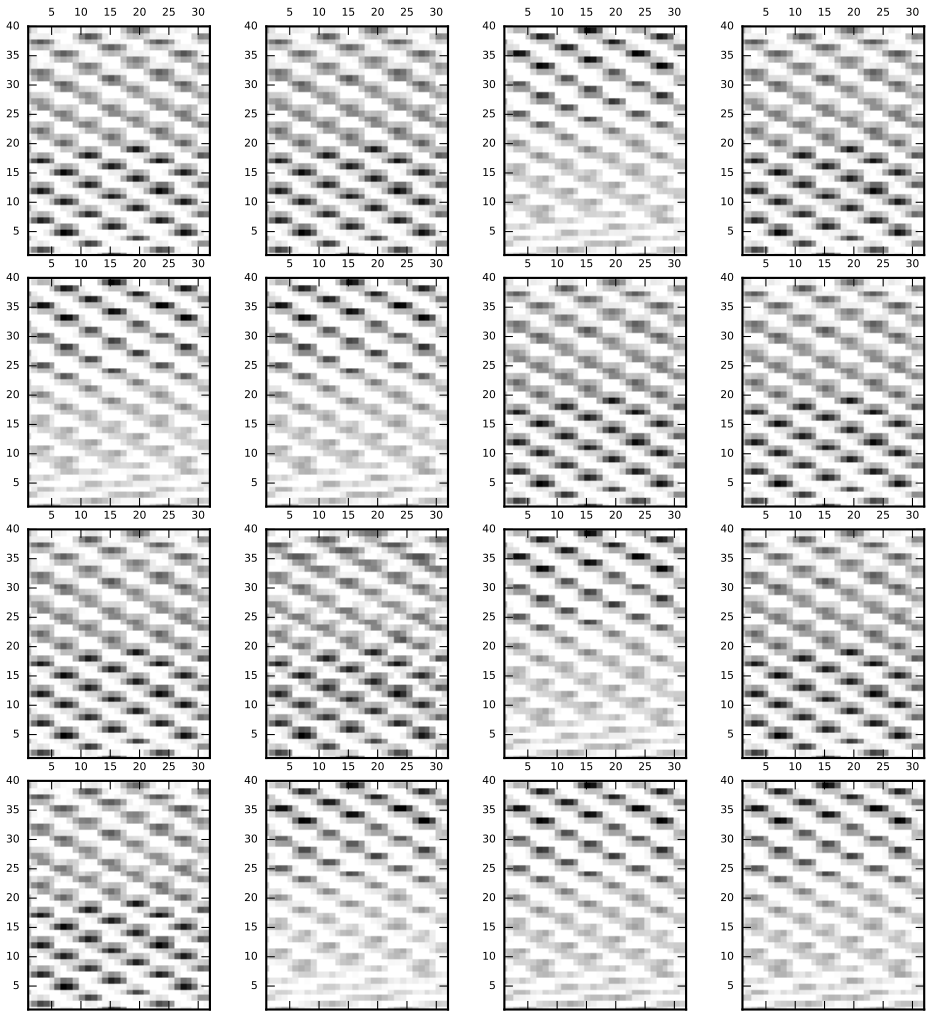

FIG. 4.5. 16 output images of the second convolutional layer after the max-pooling operation.

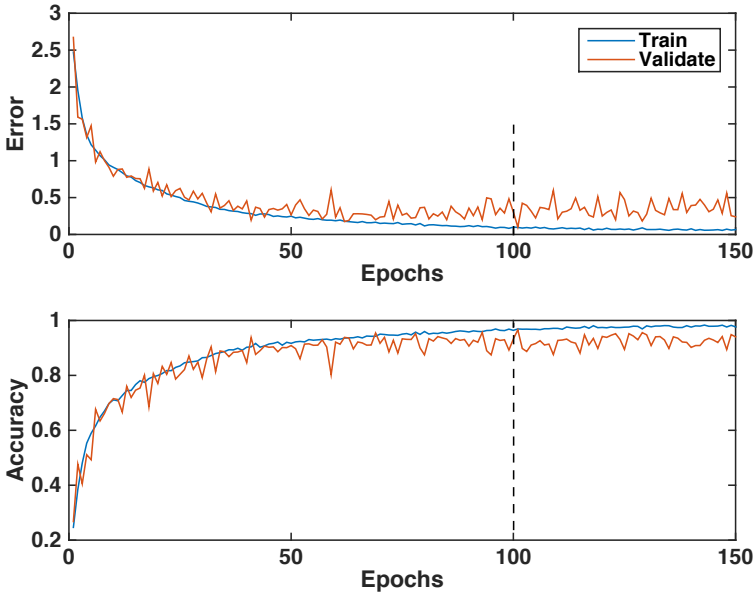

FIG. 4.6. Error and accuracy of the training and validation sets for Physical Activity Monitoring case study over 150 epochs. The vertical dashed line indicates the determined epoch.

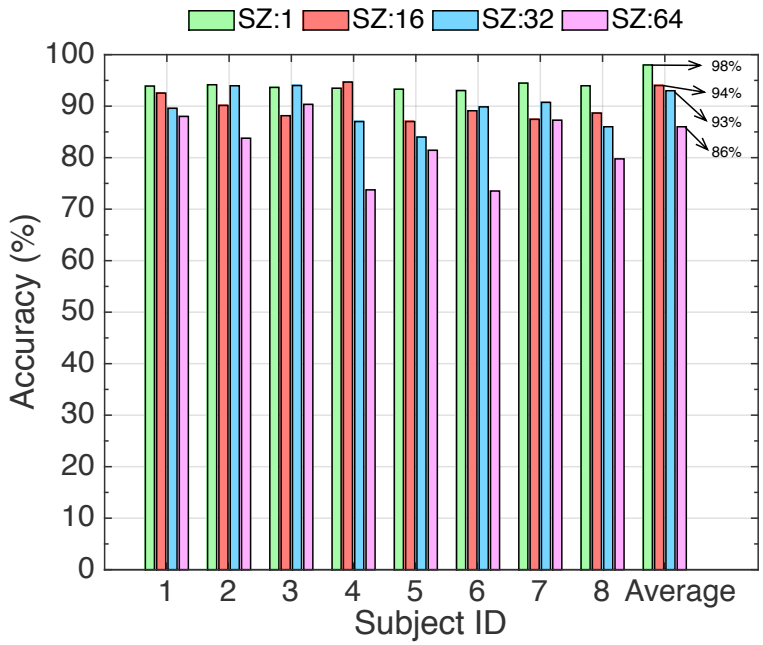

FIG. 4.7. Comparison of SensorNet classification accuracy for Physical Activity Monitoring case study. The results are for different subjects with a sliding window of size 64 samples and step-size (SZ) of 1-16-32-64.

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

4.1.5 Performance Comparison

Table 4.2 compares SensorNet performance results with state-of-the-art for Physical Activity Monitoring. [Zheng *et al.*2014] proposed an architecture which employs a CNN

per modality (variable) that processes each variable separately and then correlates them using a fully-connected layer. [Guan & Ploetz2017] proposed an architecture for Activity Recognition which uses 3 convolutional layers and instead of a fully-connected layer, they use a LSTM (Long Short Term Memory) layer and finally a softmax layer. SensorNet achieves the highest detection accuracy compared to other works for the PAMAP2 dataset. SensorNet detection accuracy is around 97.9% while [Zheng *et al.*2014] and [Guan & Ploetz2017] report 93.3% and 85.4%, respectively.

Table 4.2. SensorNet performance results comparison with existing works for Physical Activity Monitoring

Specifications	**[Zheng *et al.*2014]**	**[Guan & Ploetz2017]**	**This work**
Year	2014	2016	2017
Method	Multi-channel DCNN	Deep LSTM	SensorNet
# Subjects	7	8	8
# Activity	4	12	12
Detection Accuracy (%)	93.3	85.4%	97.9

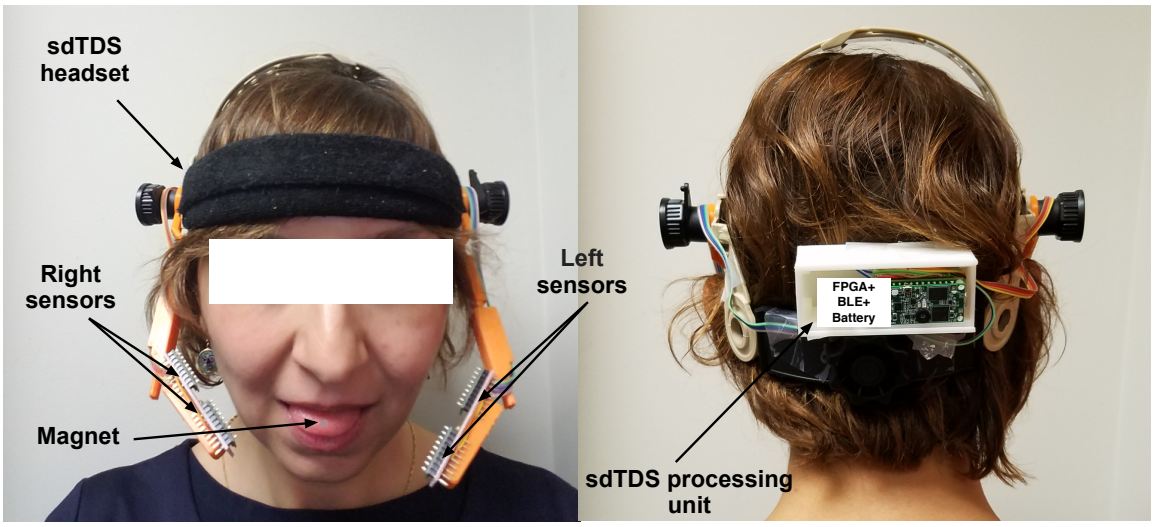

FIG. 4.8. sdTDS prototype placed on a headset which includes a low power FPGA, four acceleration and magnetic sensors, a Bluetooth low energy transceiver, a battery and a magnetic tracer which is glued to the user's tongue.

4.2 Case Study 2: Stand-alone Dual-mode Tongue Drive System

4.2.1 Overview

In this section, we propose a stand-alone dual-mode Tongue Drive System (sdTDS) shown in Figure 4.8 which uses SensorNet. sdTDS 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 and head movements. SensorNet is employed in the sdTDS to perform the entire signal processing to convert raw Magnetometer and Accelerometer sensors signals to user-defined commands, on the sdTDS wearable headset, rather than sending all raw data out to a PC or smartphone.

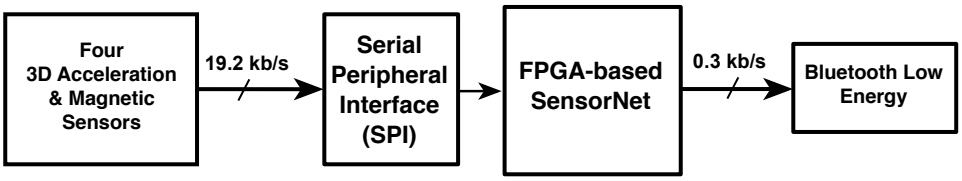

FIG. 4.9. Block diagram of the proposed sdTDS containing SPI interface, SensorNet and Bluetooth Low Energy

Fig. 4.8 shows the 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 on a box at backside of the headset. Also, the box is used for placing a battery. The box is designed using 3D printing technology and the weight of the box is around 0.14 lb.

In order to generate user-defined commands, the user should move his/her tongue to 6 specific teeth or move his/her head to 4 different directions. Any of these movements will be sensed by the sdTDS headset. Then the raw data generated by 4 magnetometers and accelerometers are transferred into SensorNet. The entire signal processing including feature extraction and classification is performed by SensorNet which can detect the user-defined commands. Figure 4.9 shows block diagram of the proposed sdTDS containing SPI interface, SensorNet and Bluetooth Low Energy

4.2.2 Motivation

Assistive Technologies (ATs) help people with severe disabilities to perform many daily activities with minimal or no assistance and increase their independence [Alam & Hamida2014, O'Brien & Ruairi2009, Kowtko2012, Genari *et al.*2013, Pawluk, Adams, & Kitada2015, Ross2001, Sarji2008, Sartori *et al.*2012, Seelye *et al.*2012, Leonardis *et al.*2015, Cook *et al.*2005, Matsubara *et al.*2015, Van Erp, Lotte, & Tangermann2012, Horki *et al.*2015, Caltenco *et al.*2012, Mekhalfi *et al.*2015]. These systems allow the users to send commands to an external device, such as a motorized chair, smart phone or computer. In order to allow those with severe disabilities to interact with the environment around them, different ATs have been developed that use different sensor modalities such as electroencephalogram (EEG) [isc, McFarland & others2008] and electrooculogram (EOG) [Wolpaw & others2002], eye movements [Barea & others2002], head motion [Pereira & others2009], and facial muscle activity [Huang & others2006].

Paralyzed individuals with severe physical disabilities, such as those with spinal cord injury (SCI), traumatic brain injuries (TBI), and some type of stroke, who suffer from tetraplegia, a condition in which all four limbs are paralyzed, heavily rely on ATs to enhance their quality of life and to live more productively and independently [Carlson & Ehrlich2005].

Because of following reasons, the tongue is an ideal source of volitional commands for developing a wearable AT system for people with severe paralysis [Kandel &

others2000]: 1) Sophisticated motor control capability evident in speech and ingestion, 2) Fast movement with many degrees of freedom and very flexible, 3) It is connected to the brain by a cranial nerve: it escapes even high level spinal cord injuries, 4) Noninvasive access to tongue is possible, 5) it is not afflicted by repetitive motion disorders, 6) it does not fatigue easily [Krishnamurthy & Ghovanloo2006], 7) It is all inside the mouth and it has privacy advantage, 8) it is not influenced by the position of the rest of the body [Viseh, Ghovanloo, & Mohsenin2015].

Several tongue-computer interfaces have been proposed in recent years [Lau & OLeary1993, Struijk2006, Lund & others2009, NS Andreasen Struijk & others2016, Nam & others2012, Saponas & others2009, Zhang & others2015, Huo, Wang, & Ghovanloo2008, Yousefi *et al.*2012, Huo *et al.*2013]. Typically, these systems capture signals from analog sensors, digitize them after signal conditioning and send all the raw data through a wireless transmitter to a receiver platform for further processing, such as feature extraction and classification. The receiver platform can be a computer or smart phone. In the case of TDS, assuming Analog Front End (AFE) takes at least 50 samples per second [Zhang & others2015] for each of the four sensors and each sample has X, Y, and Z data which is 16 bits each, the transmitter needs to send 9.6 kbit/s to the receiver side. Therefore, the drawback of this type of system is that constantly transmitting this amount of raw data using a transmitter such as Bluetooth Low Energy (BLE) results in high power consumption [Jafari *et al.*2015].

4.2.3 Related Work

In recent years, several wearable tongue drive ATs for people with severe disabilities have been developed which are discussed in this section. In [Huo & others2007] authors proposed a TDS for controlling a mouse. In their work, External Magnetic Interference (EMI) effects have been reduced to an acceptable level by adding a reference 3-D compass. Principal component analysis (PCA) and k-Nearest Neighbor (KNN) algorithms are used to associate the magnetic field sensor outputs to 6 different direct mouse commands. In [Yousefi *et al.*2012] authors evaluated a tongue operated AT as a switch-based pointing device with four directional commands for computer access and achieved an accuracy of 94.7%. Zhang et al. [Zhang & others2015] presented a new rehabilitation robot, called Hand Mentor (HM) Pro^{TM}, which reads its pressure and joint angle sensors, combined with control commands from the TDS to enable both isometric and isotonic target-tracking tasks in a coordinated tongue-hand rehabilitation paradigm. In [Huo *et al.*2013, Sahadat & others2015] authors introduced a multi-modal version of TDS, which takes different sensor modalities such as tongue motion, speech and head tracking. They used the system for controlling a mouse, typing and sending an email. [Huo *et al.*2013] reported an overall 85.1% $\pm$ 8.8% recognition accuracy. In all these works, raw data needs to be transmitted wirelessly to a receiver platform for further processing, which results in high power consumption. Also, they depend on the receiver platform to run a software for signal processing, such as MATLAB and LabVIEW. In [Saponas & others2009], authors

presented an optical tongue drive system with an accuracy of 92%. Their proposed prototype is wired, which is not convenient for a paralyzed patient. In their work, each gesture takes 1.5 seconds on average to perform and recognize which is a high latency for real-time applications such as wheelchair driving. Authors in [Nam & others2012] present a tongue machine interface based on using Glossokinetic potentials (GKPs) which are electric potential responses generated by tongue movement. They use the proposed system for controlling a wheelchair. Their system requires the users to carry a scalp on their head in order to record EEG signals and only detects and uses three commands for wheelchair control. In [Viseh, Ghovanloo, & Mohsenin2015], authors proposed a TDS local processor with an accuracy of 93.3% which performs all signal processing on the sensor side and send out only the detected commands. The results were based on Verilog simulations and actual hardware was not built and tested. KNN was used as a machine learning classifier which consumes high energy due to requiring many computations and a large on-chip memory to store the classifier's training data. Authors in [Lund & others2009, NS Andreasen Struijk & others2016] present a wireless, intraoral and inductive tongue computer interface to type using the keypad and mouse pad area. Their proposed system, does not need a receiver platform such as computer/smartphone to run a software for processing. It is worth mentioning that, the intraoral [Lund & others2009, NS Andreasen Struijk & others2016] and the headset [Huo *et al.*2013, Sadeghian, Huo, & Ghovanloo2011] versions of a tongue-operated AT each have their pros and cons. Therefore, the ultimate choice depends on the preference of the end user, i.e. whether they prefer comfort over aesthetics

or vice versa. In fact, their preference might even depend on the environment that they are in. For instance, they might choose to use the headset version at home, but wear the intraoral version outside or in social events. Nonetheless, both versions will immensely benefit from conducting the classification on the sensor side.

Compared to the previous works, in [Jafari *et al.*2017], we developed a stand-alone Tongue Drive System (sdTDS) which performs the entire signal processing at the sensor node which is a light-weight headset and only sends out the final decision (3 bits) through a BLE. The BLE doesn't need to be active at all, as it sends these 3 bits through advertising packet [Gomez, Oller, & Paradells2012]. Therefore, the power consumption due to the wireless transmission is reduced significantly. Furthermore, the proposed system does not depend on a receiver platform to run a software for processing, such as MATLAB and LabVIEW. Hence, a user does not need to have another device other than a headset which makes the user more independent. Also, employing the proposed stand-alone system reduces the cost and system failure rate due to having multiple processing units. Figure. 4.10 shows a comparison between the power consumption of TDS when sending all raw data out versus preforming the processing locally and sending out only the decisions. As it can be seen from the figure, the power consumption of sTDS is significantly lower than TDS. TDS consumes around 21.6 mW power for receiving and sending 9.6 kbit/s raw sensor data, whereas sTDS consumes 8.8 mW power for receiving, processing and issuing detected commands. Also, the proposed system is not dependent on a receiver platform to run a software for processing, such as MATLAB and LabVIEW. Therefore, a user does not

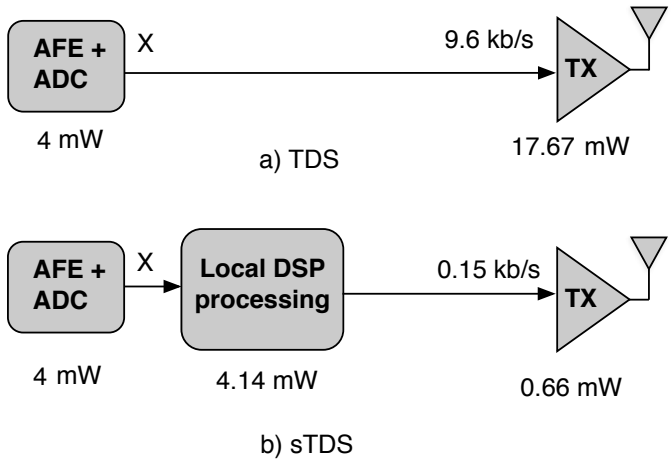

FIG. 4.10. a) TDS: Sending all raw data out. b) sTDS: performing all processing locally and sending out the decision. The transmission power is reduced significantly by performing all signal processing locally on the sensor side. Measured power consumption numbers are shown separately for 1) AFE which generates 9.6 kbit/s raw magnetic data, 2) Local DSP processor on a low power FPGA and 3) Bluetooth low energy.

need to have another device other than a headset which makes the user more independent.

In [Jafari *et al.*2017], to validate functionality of sTDS in the testing phase, a computer-based Maze navigation game is designed and tested. Fig. 4.11 presents a GUI which shows the Maze game, elapsed time of the game, command positions on mouth [Huo *et al.*2013] and different buttons which are controlled by sTDS. A user should navigate the Maze using the commands generated by their tongue. The goal is to move from the start to the end point, which is a star. In order to generate the commands, the user should move his/her tongue to the specific teeth which they used previously in the training phase. The

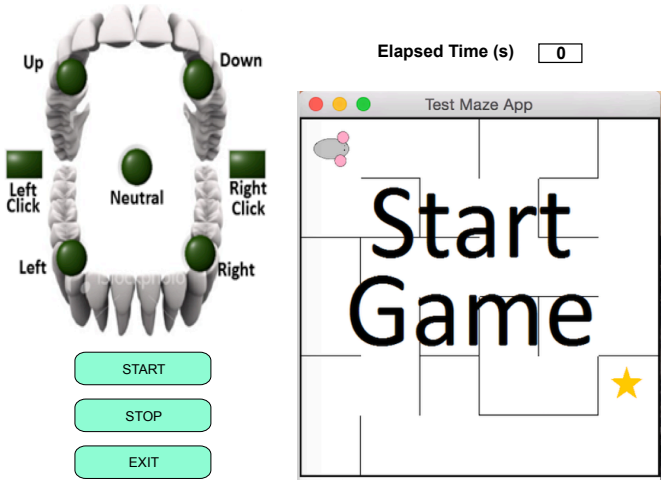

FIG. 4.11. Maze navigation game GUI with a start and end location. During the experiment a small mouse automatically moves to start location. User starts to move mouse using 4 different tongue commands (left, right, up and down) to navigate from start to end.

goal of the experiment is to finish the game as fast as possible from start to end using the sTDS. Four users (age: 26-37, 3 male and 1 female, experienced-familiar) played the Maze game five times. The results showed that all users finished each round of the game in less than two minutes.

4.2.4 Experimental Setup and Results

The front end of the system is composed of four magnetic and acceleration sensors (LSM303D), which have onboard 3D compasses that are used to monitor the magnetic field generated by a magnet tracer which is glued to user's tongue [Mimche *et al.*2016] and the same time head movement. Each sensor is configured to provide a magnetic field full-scale of +/- 8 Gauss. The interface to the AFE uses two SPI buses, one for the left

two sensors and one for the right two sensors. Each sensor can be configured to use one of two possible slave addresses, allowing two sensors on the same bus. In order to minimize the device utilization needed for the interface, only one instance of the SPI bus master was instantiated and it is used to read each of the sensors one at a time. Each sensor provides a 16-bit two's complement reading for each of its three axes. Once all four sensors have been read, the interface then passes the data to SensorNet.

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 Chapter 3 Section 3.3, 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 set, 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. SensorNet detection accuracy for the sdTDS is 96.2%.

4.2.5 Performance Comparison

Table 4.3 compares SensorNet performance results with state-of-the-art for Tongue Drive Systems. As we explained earlier, in [Jafari *et al.*2017] we proposed a single-mode stand-alone Tongue Drive System which only detect the tongue movement. We designed an External Magnetic Interference (EMI) attenuation algorithm to attenuate the effect of the external magnetic interference on the raw data and also we used Logistic Regression (LR) for classification. The sTDS detection accuracy is around 96.9%. In [Jafari, Ghovanloo, & Mohsenin2017], we proposed a dual-mode Tongue Drive System which can detects user's tongue motion using a magnetic tracer placed on tongue and an array of magnetic sensors embedded in a wireless headset and at the same time it can capture the user's voice using a small microphone embedded in the same headset. We used EMI and LR for the tongue movement detection and Cross-correlation for Speech Recognition. The detection accuracy is 96.6% for tongue motion, and 97.5% for speech recognition. In this dissertation, SensorNet is employed to detect the tongue and head movements. Although the number of input channels and activities are higher in this work, SensorNet achieves similar accuracy compared to [Jafari *et al.*2017, Jafari, Ghovanloo, & Mohsenin2017] which is around 96.2%.

Table 4.3. SensorNet performance results comparison with existing works for sdTDS

Specifications	**[Jafari *et al.*2017]**	**[Jafari, Ghovanloo, & Mohsenin2017]**	**This work**
Year	2017	2017	2017
Modality	Tongue	Tongue & Voice	Head & Tongue
# Data Channels	12	13	24
# Activity	7	11	12
Assistive Technology	sTDS	sdTDS	sdTDS
Technique	EMI & Logistic Regression	EMI & Logistic Regression & Cross-correlation	SensorNet
Detection Accuracy (%)	96.9	96.9 & 97.5%	96.2

4.3 Case Study 3: Stress Detection

4.3.1 Dataset Description

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 [Birjandtalab *et al.*2016]. 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.

4.3.2 Experiment Setup

As it was discussed in Chapter 3 Section 3.3, the 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 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

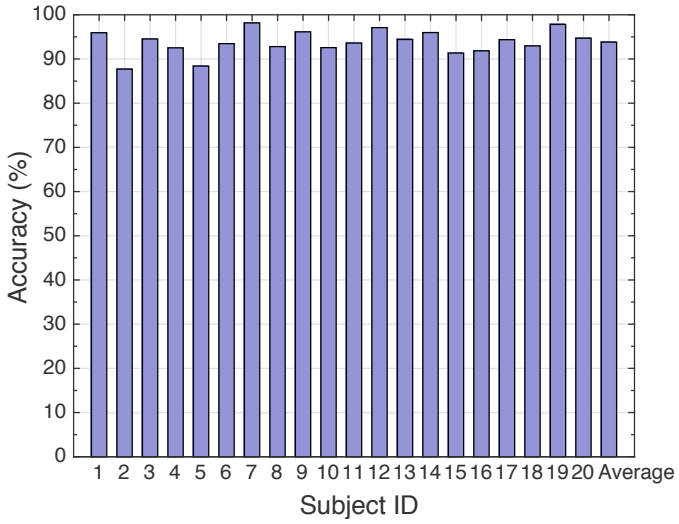

FIG. 4.12. SensorNet classification accuracy for Stress Detection case study. The results are for 20 different subjects and average accuracy over all 20 subjects is around 94%.

satisfactory. Therefore, for all the experiments for this dataset we train SensorNet with 100 epochs.

4.3.3 Experiment Results

Fig. 4.12 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%.

4.3.4 Performance Comparison

Table 4.4 compares SensorNet performance results with state-of-the-art for Stress Detection case study. [Birjandtalab *et al.*2016] proposed an architecture which employs Gaussian Mixture Model (GMM) which is an unsupervised clustering technique. They could separate different neurological status with an accuracy of approximately 85%. Compared to [Birjandtalab *et al.*2016], SensorNet achieves 9% higher detection accuracy.

Table 4.4. SensorNet performance results comparison with existing works for Stress Detection

Design	**[Birjandtalab *et al.*2016]**	**This work**
Year	2016	2017
Technique	Gaussian Mixture model	SensorNet
# Subjects	20	20
# Activity	4	4
Detection Accuracy (%)	84.6	94

4.4 SensorNet Optimization Evaluation

As it was discussed in Chapter 3 Section 3.3, 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 logic behind of choosing the SensorNet architecture and parameters.

One of the primary objective of this dissertation is to be able to efficiently deploy SensorNet in embedded and resource-bound settings which is very challenging because of strict power and area budgets in these settings. 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 sizes, 4) Input zero-padding, 5) Activation functions, and 6) Pooling layers

4.4.1 Number of Convolutional Layers

In this experiment, we compare six SensorNet designs with an increasing number of convolutional layers, for the three different case studies. These 6 configurations are depicted in Figure 4.13. The comparison has been made in terms of detection accuracy, number of convolutional operations, number of parameters (model weights) and memory requirements. 64-bit resolution is used to calculate the memory requirements.

Figure 4.14 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 improves slightly but for the Stress Detection reduces because the useful features may be filtered out during the convolutional and max-pooling processes.

Figure 4.15 depicts the impact of increasing the number of convolutional layers on the number of model parameters and memory requirements. As is shown in the figure, by increasing the number of convolutional layers, the 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. However, by adding additional

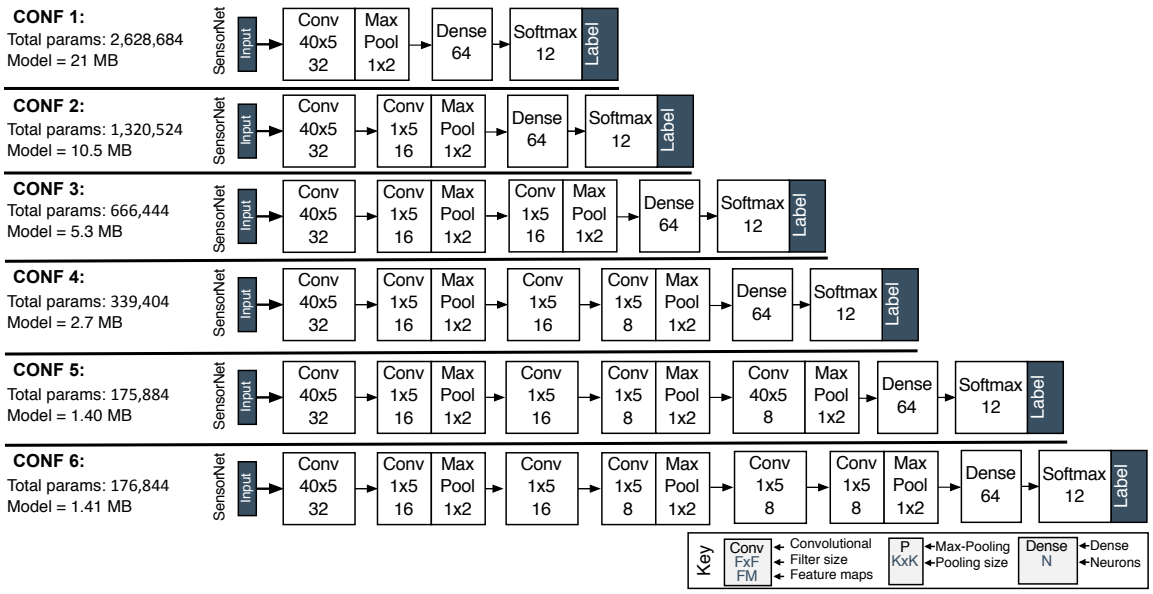

FIG. 4.13. A comparison of number of required parameters (model weights) for different SensorNet configurations for Physical Activity Monitoring application. Each configuration has different number of convolutional layers and consequently different model size. Model weights includes the parameters for convolutional, fully-connected and Softmax layers. 64-bit resolution is used to calculate the memory requirements. Configuration 5 has the minimum number of model weights.

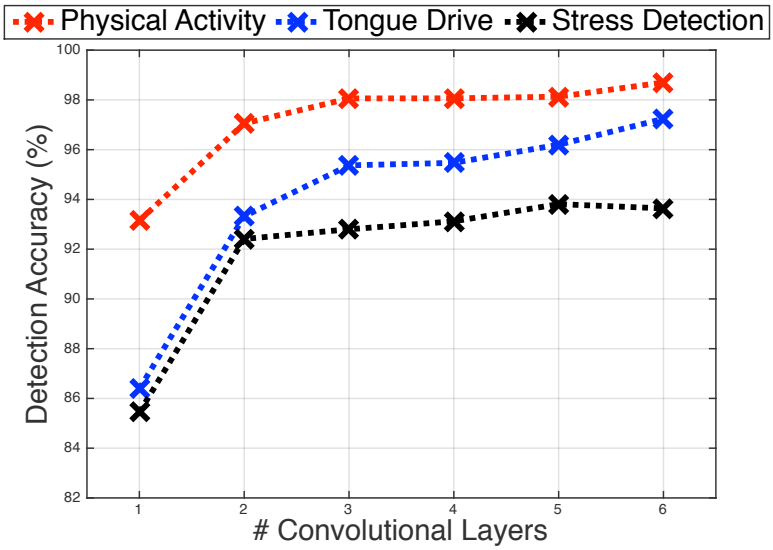

FIG. 4.14. Impact of increasing the number of convolutional layers on SensorNet detection accuracy for Physical Activity Monitoring , sdTDS and Stress Detection case studies.

convolutional layer the number of floating-point operations to finish a classification task increases slightly which is shown in Figure 4.16. This analysis results show that a SensorNet with 5 convolutional layers is the best candidate with regards to detection accuracy, number of convolutional operations and memory requirements.

4.4.2 Number of Filters

The number of filters (weights) are another important hyperparameter for implementing SensorNet on a low-power and resource-limited embedded device 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

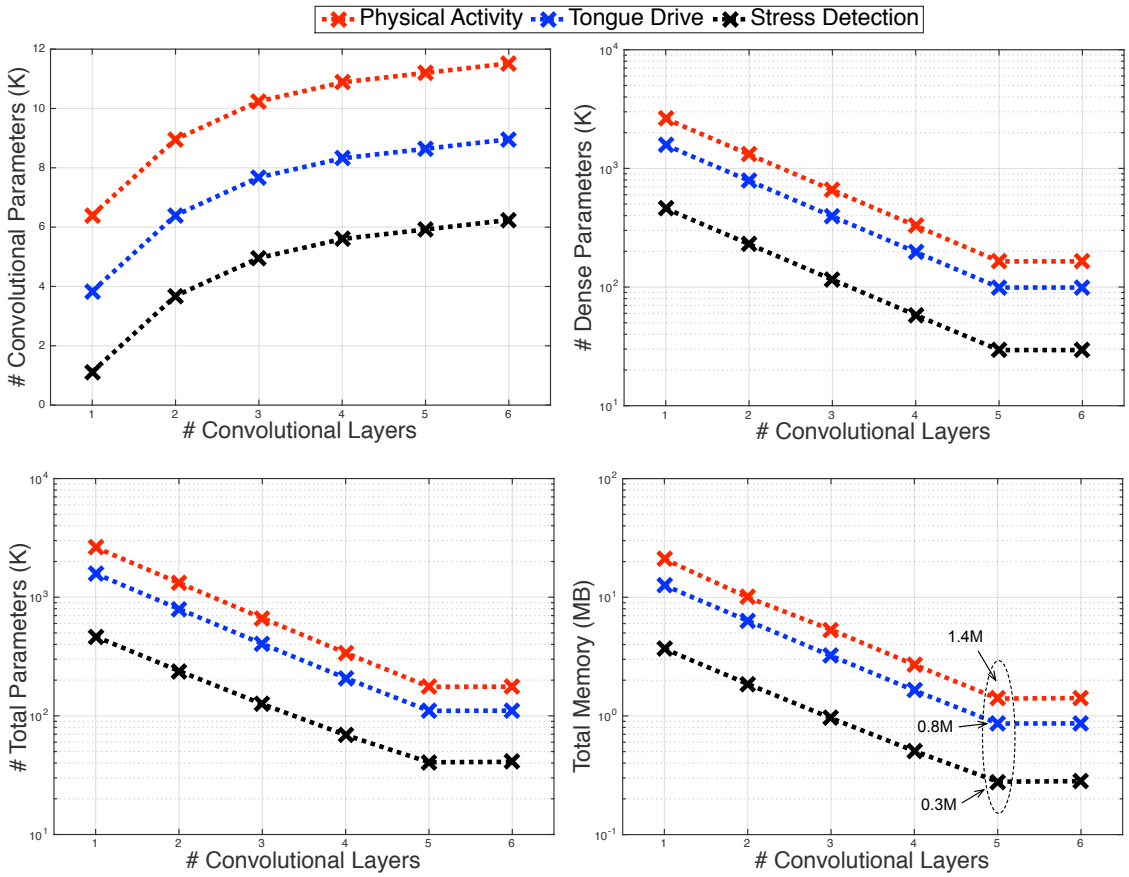

FIG. 4.15. Impact of increasing the number of convolutional layers on floating-point operations, memory requirements of SensorNet, for three different applications. By adding additional convolutional layer, memory requirements reduce dramatically.

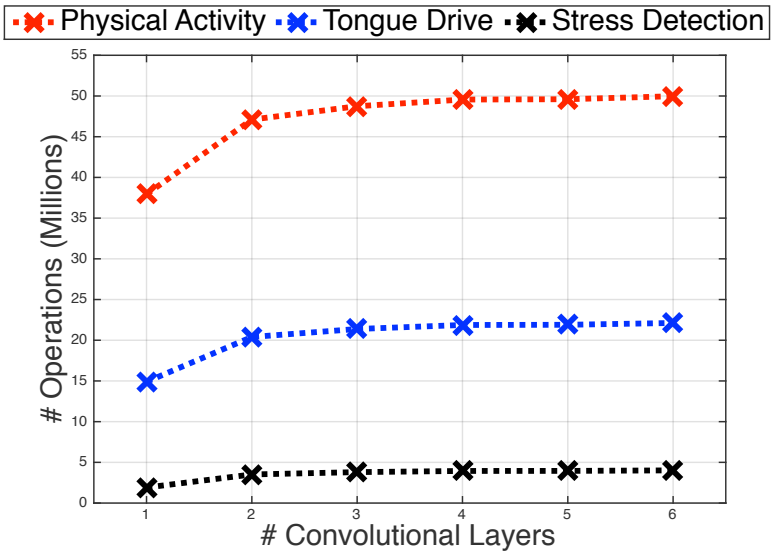

FIG. 4.16. Impact of increasing the number of convolutional layers on floating-point operations for Physical Activity Monitoring, sdTDS and Stress Detection case studies. By adding additional convolutional layer the floating-point operations increases slightly.

a direct effect on energy consumption. In this experiment, we keep the number of convolutional layers fix (5 layers) and increase the number of filters for each layer as is shown in Figure 4.17. 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. Figure 4.18 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. For example, for the Physical Activity Monitoring, Set 1, Set 2, Set 3 and Set 4 need 1.4 MB, 2.88 MB, 6 MB and 7.8 MB memory to save model weights with a detection accuracy of 97.9%, 99%, 99.2% and 99.45%, respectively. Based on the results, all the configurations achieve similar detection accuracy but Set 1 needs much lower number of parameters. and therefore Set 1 is chosen to be implemented on hardware.

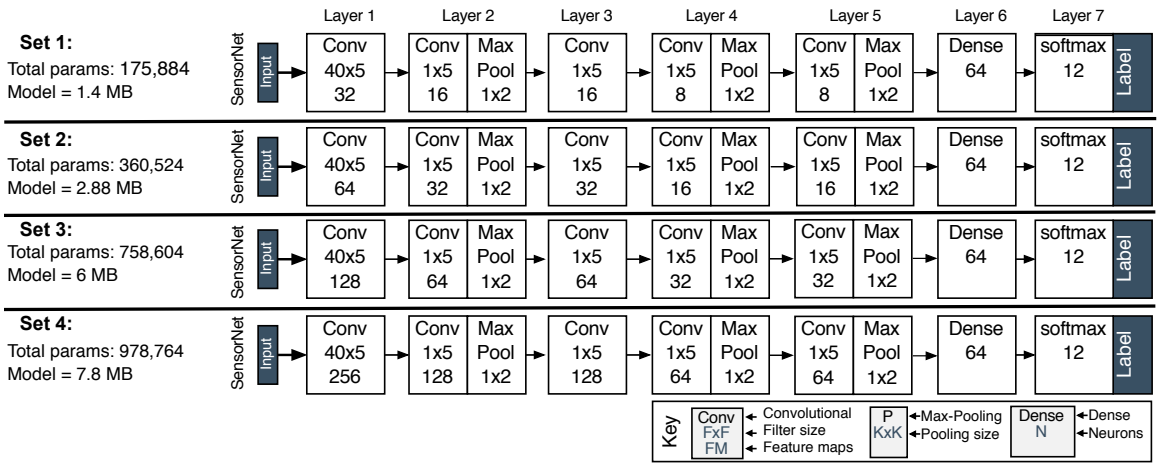

FIG. 4.17. A comparison of number of required parameters (model weights) for different trained models when we increase the number of filters for each layer, for Physical Activity Monitoring case study. Model weights includes the parameters for convolutional, fully connected and Softmax layers. Set 1 has the minimum number of model weights.

4.4.3 Shape of Filters

Another important parameter for implementing SensorNet on an low-power embedded platform is the filter shape. As it was explained in Chapter 3 the idea is to generate some shared features across 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

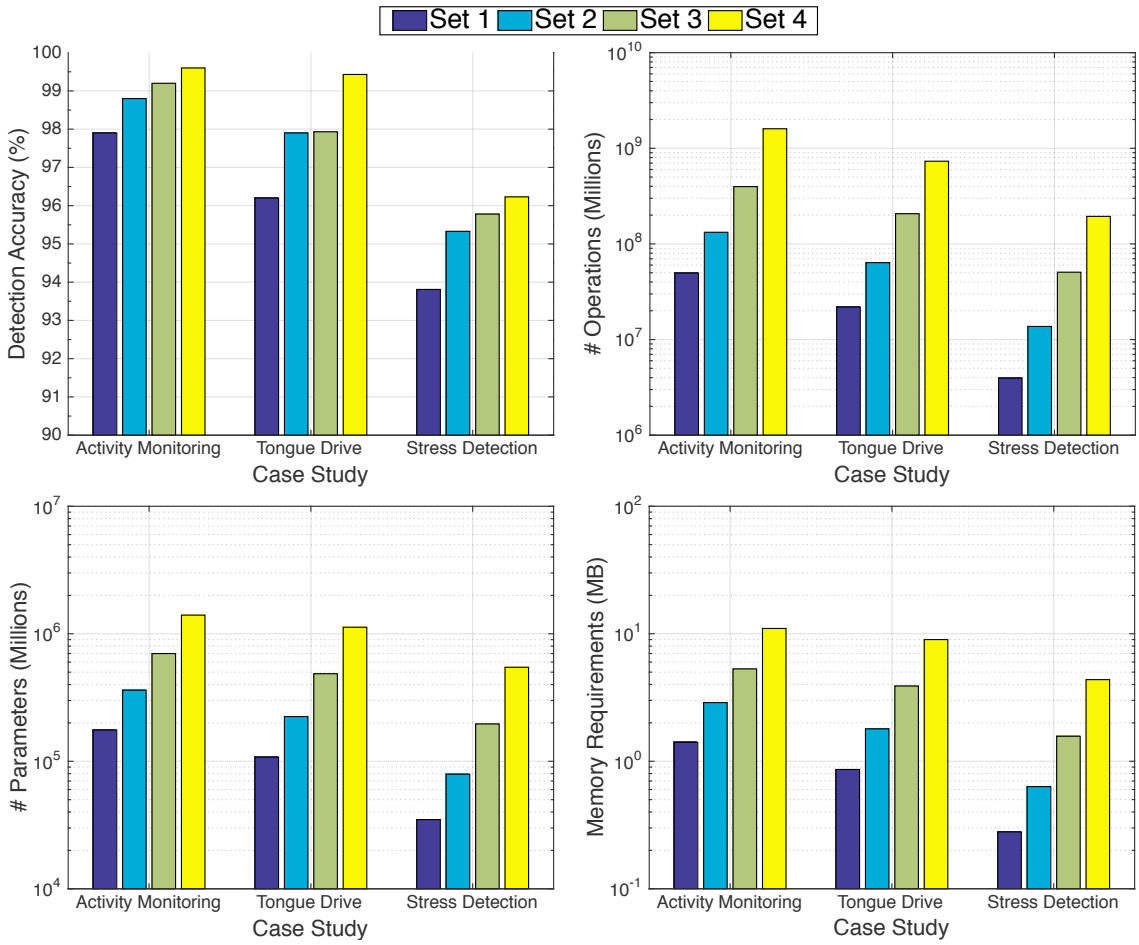

FIG. 4.18. Comparison of four different SensorNet configurations in terms of floating-point operations, memory requirements and detection accuracy. By adding additional model weights to each layer, the computation and memory grows dramatically with only modest improvement in detection accuracy.

layer takes 1-D vectors rather than images. Furthermore, it reduces the the number of operations which this affects directly on energy consumption of the framework when is implemented on an embedded platform.

In this experiment we change the filters size for the first convolutional layer to 5×5, 3×3 and 1×5 for Physical Activity Monitoring, sdTDS and Stress Detection case studies. Based on the results, filter size of $M \times 5$ (M is the number of input modalities) gives better detection accuracy compared to 5×5, 3×3 and 1×5 filter sizes. This is shown in Figure 4.19. Also, another interesting finding is that, for the dataset with more number of input channels, choosing $M \times 5$ 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.

4.4.4 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 applications. 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 Figure 4.20, 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.

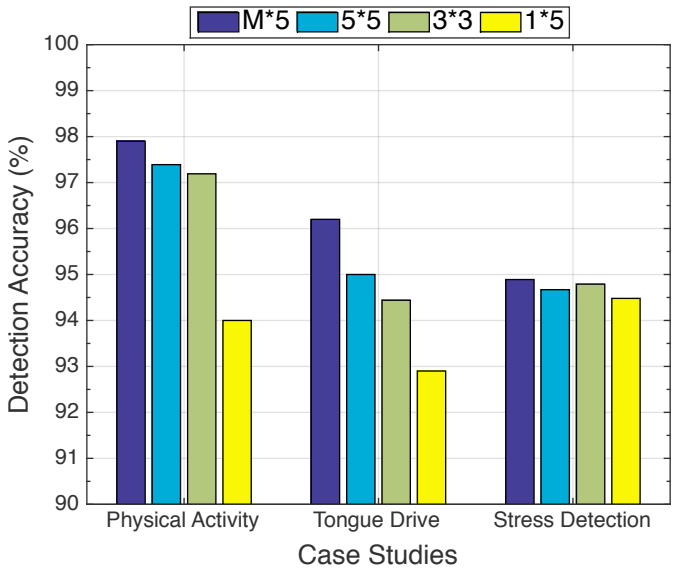

FIG. 4.19. Comparison of SensorNet detection accuracy using four different filter sizes.

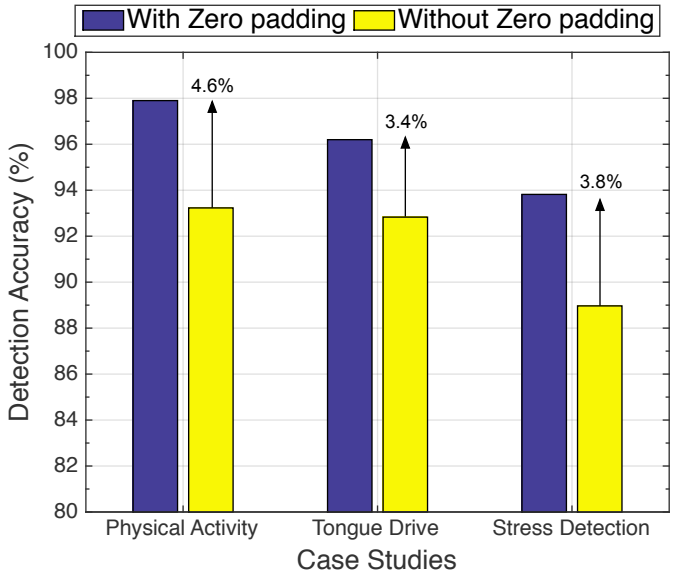

FIG. 4.20. Impact of zero-padding on SensorNet detection accuracy for Physical Activity Monitoring, sdTDS and Stress Detection case studies, respectively.

4.4.5 Activation Functions

In Chapter 3 Section 3.1, we mentioned that ReLU activation function is efficient because they require 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 the 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 Figure 4.21. 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 detection accuracy.

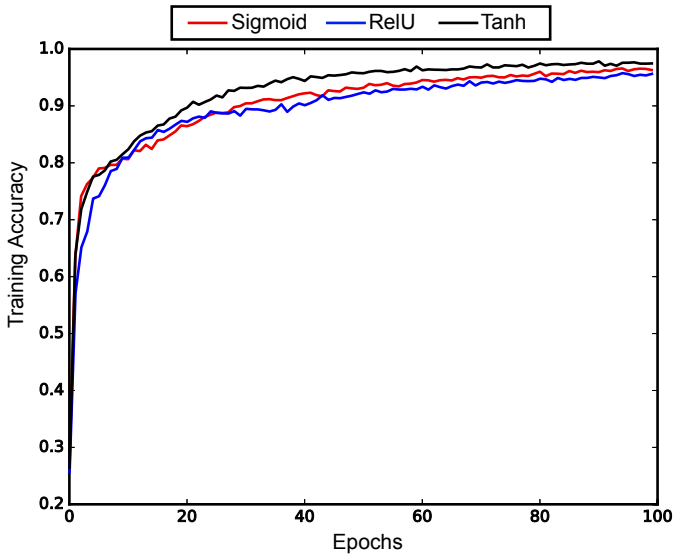

FIG. 4.21. Impact of different activation functions including Sigmoid, Tanh and ReLU in the fully-connected layer on the SensorNet detection accuracy. The results are based on the Tongue Drive System case study.

Chapter 5

PROPOSED HARDWARE ARCHITECTURE

Another major contribution of this dissertation is the development of a hardware architecture for efficient deployment of SensorNet on IoT and wearable devices which need to work in real-time, must consume low power and are resource-limited. Following are the main objectives for the hardware architecture design:

- Consumes minimal power
- Meets latency requirement of an application
- Occupies small area
- Needs to be fully reconfigurable
- Requires low memory

5.1 Design Methodology

We propose a fully serial hardware architecture for SensorNet. A serial architecture will consume the least amount of power. However, the challenge is the latency to perform all the required computations to finish a classification task. We don't need a high throughput for SensorNet as the input data are time series and the sampling frequencies are not high usually, but it has to meet a minimum deadline which we assume a user issues one command every 1 second. Therefore, we design SensorNet to have very low power consumption at expense of throughput and latency. Also, we design SensorNet hardware architecture to be reconfigurable, because different applications have different requirements. Parameters such as filter sizes in the convolutional layers, zero-padding, sizes of the fully-connected and softmax layers are configurable. Another challenge is convolutional layer memory management. To save resources such as memory, we design convolutional module to read from the memory and write back the results into the same memory cell. This will save memory requirement drastically.

5.2 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 both on memory requirements and also on power consumption. We model a custom fixed-point SensorNet design to find optimum weights length. To quantify performance gap between floating

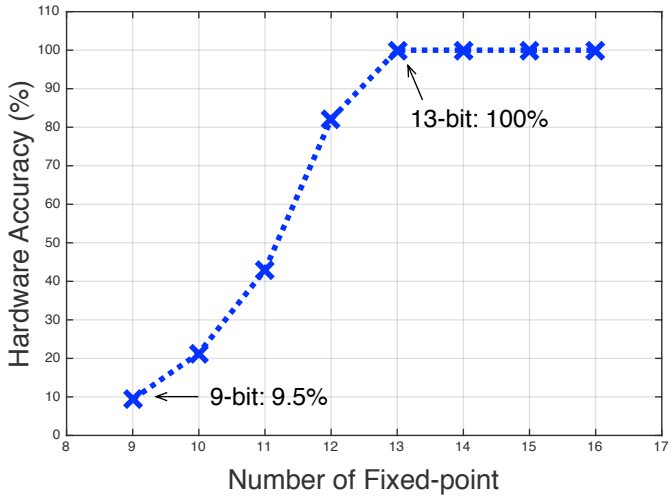

FIG. 5.1. Hardware accuracy of SensorNet with respect to number of fixed-point bits used to represent the filters.

point Keras implementations and proposed fixed-point architectures, we calculate average accuracy from the estimated signal obtained from Python software solution and hardware implementations. Figure 5.1 shows accuracy of SensorNet computation signal is dependent on number of fixed-point bits used to represent the weights. Based on the results, 3.13-bit fixed-point format gives hardware accuracy of 100% with an error of 2ˆ13. Therefor, all SensorNet filters are converted to 3.13-bit (16-bit) fixed-point format and saved in the memory.

5.3 Hardware Architecture

Implementing hardware architecture for SensorNet faces several challenges such as buffering of the input data, computational model implementation and managing

memory transfers. Figure 5.2 depicts SensorNet hardware architecture with circuit-level implementation details. The main components of SensorNet on hardware consists of the following:

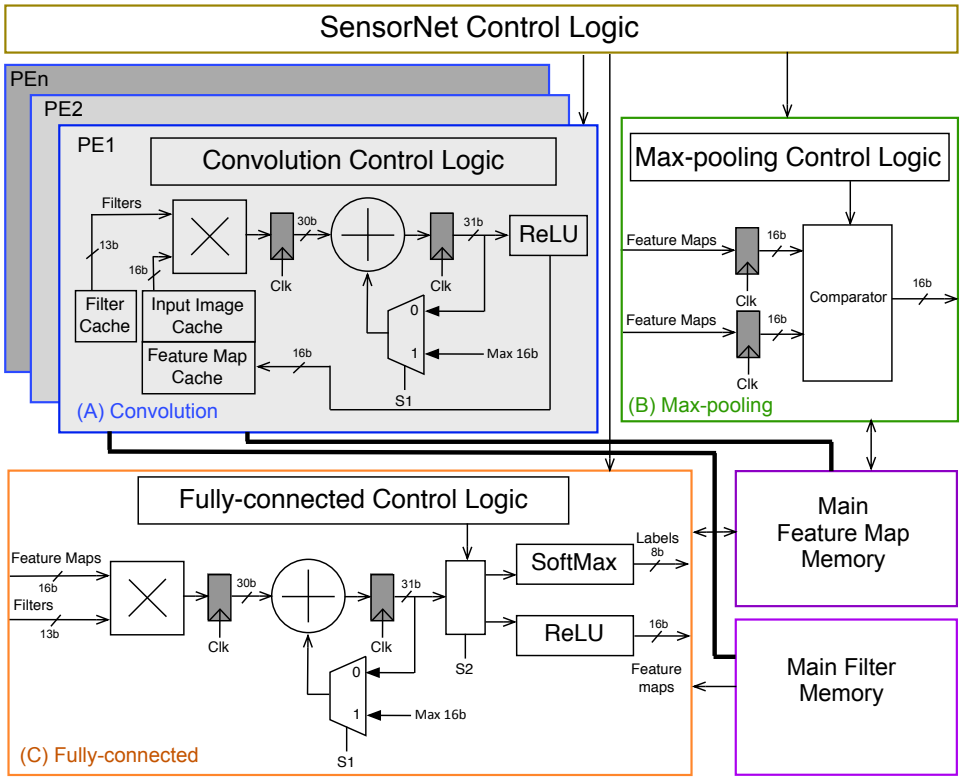

FIG. 5.2. Block diagram of SensorNet hardware architecture which includes convolution, max-pooling and fully-connected blocks and also a top-level state-machine which controls all the blocks. PE refers to convolution Processing Engine (PE)

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

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

(C) Fully-connected Performs fully-connected layer operations. The fully-connected block includes ReLU and SoftMax activations 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. 5.2-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 multiplexers, a few registers, 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. 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. A fixed-point analysis is performed to find out the best number of bit representation in each stage of the hardware architecture design which will be discussed at the end of this chapter. An offline training is performed to obtain model weights using keras. The model weights are converted to fixed-point format and are represented by 16 bits. The floating-point arithmetic is complex and requires more area, therefore use of fixed point arithmetic will avoid complex multipliers. The input to the max-pool is feature maps data, which is formed by convolution block. The max-pool block contains some registers and a comparator. 5.2-C shows the a fully-connected block. As is shown, the architecture consists of a serial dot

product engine, a state machine for controlling all sub-blocks. Also, the fully-connected block has both ReLU and Softmax activations functionality. Depends on the layer either one can be used.

5.4 Exploiting Efficient Parallelism

As we mentioned in previous section, we primary target a fully serial hardware architecture for SensorNet. However, we design SensorNet hardware architecture to be configured to perform convolution operation in parallel if it is needed. In deep convolutional neural networks, convolutional layers dominate the computation complexity and consequently affects on 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 [Page & others2017] 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 Figure 5.3. 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.*2015] analyzed these three tiling methods using the rooftop model to determine what method provides the best throughput in FPGA fabric. There

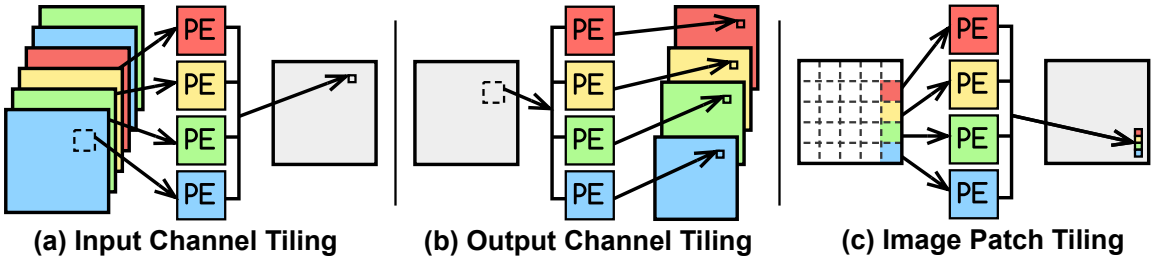

FIG. 5.3. Comparison of different parallel tiling techniques for convolutional layers. Output channel tiling has the least communication contention and inter-core dependency.

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 the high parallelism, minimal dependency among the parallel cores and minimal communication contention.

Chapter 6

HARDWARE IMPLEMENTATION RESULTS

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

6.1 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 50 MHz. Verilog HDL is used to describe architecture and hardware of the SensorNet.

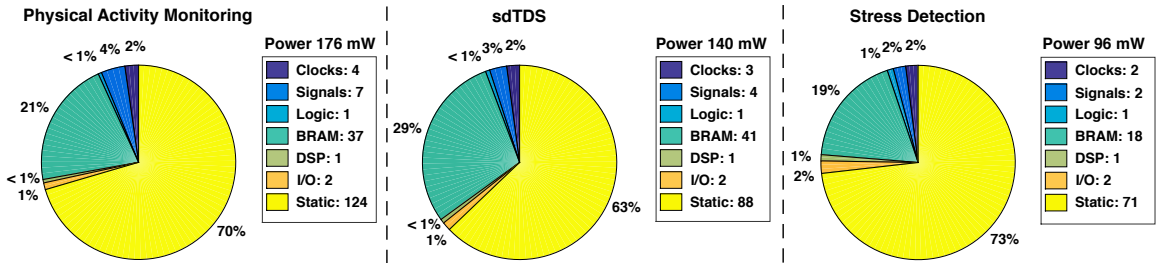

FIG. 6.1. Serial implementation power results on FPGA, for Physical Activity Monitoring, sdTDS and Stress Detection case studies.

6.1.1 Power Consumption Results and Analysis

Figure 6.2 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 table, average device static and Block RAMs power consumption of FPGA is around 69% and 23% of entire power which is very large compared to the power consumption of SensorNet logic. However, overall the power consumption is small and is suitable for battery-powered wearable ATs.

6.1.2 Resource Utilization Analysis

Table 6.1 show the device utilization of SensorNet on Xilinx FPGA for Physical Activity Monitoring, sdTDS and Stress Detection case studies. We use different packages of Xilinx FPGA Artix-7 for different applications. Small package is enough for small applications such as Stress Detection. As it can be seen from the table, SensorNet logic only utilizes small portion of the FPGA fabric but on average 70% of memory (BRAMs)

Table 6.1. Implementation results of the proposed SensorNet on FPGA. The Results obtained at clock frequency of 50 MHz

Application	**Physical Activity Monitoring**			**Tongue Drive System**			**Stress Detection**		
Device	Xilinx Artix-7 (XC7A200T)			Xilinx Artix-7 (XC7A75T)			Xilinx Artix-7 (XC7A35T)		
Resource	Used (#)	Available (#)	Utilization (%)	Used (#)	Available (#)	Utilization (%)	Used (#)	Available (#)	Utilization (%)
DSP Slices	3	740	<1	3	180	<1	3	90	3.3
BRAM	176	365	**50**	96	105	**92**	32	50	**64**
Slice	525	33650	<1	96	15850	<1	295	8150	3.6
Slice LUT	1285	134600	<1	927	47200	2	631	20800	3
Slice Registers	420	269200	<1	399	94400	<1	412	41600	1

are utilized for filters and feature maps.

6.1.3 Parallel Implementation Results

Scalability is one of the key features of the proposed SensorNet on hardware. Although we proposed a fully serial architecture for SensorNet, the architecture is configurable and can be parallelized based on any application specifications. In this section, we evaluate the impact of increasing the number of convolutional processing engines (PE) on classification throughput, energy, and area utilization. Each PE contains one convolutional block with ReLU activation function. Table 6.2 provides SensorNet performance results for different architectures, including serial, semi-serial and fully-parallel designs. Also, Figure 6.3 demonstrates the impact of increasing the number of PEs for three different case studies. As is shown, for a given network increasing the amount of PEs can

Table 6.2. Implementation results of the proposed SensorNet on FPGA. The Results obtained at clock frequency of 50 MHz

Specifications/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
# Used PE	1	4	8	1	4	8	1	4	8
latency (S)	659	168	85	444	113	57	225	56	28
Throughput (label/S)	1.4	5.4	10.6	2	8	16	4	16	31.4
Dynamic Power (mW)	51	60	72	51	60	72	25	34	46
Static Power (mW)	124	124	124	88	88	88	71	71	71
Total Power (mW)	175	184	196	139	148	160	96	105	117
Total Energy (mJ)	115	30	16.8	62	16	9	21	6	3.5

improve both throughput and energy consumption. Therefore, if an application needs to perform a classification task very fast, parallelism is necessary.

6.1.4 Impact of Zero-padding on FPGA Results

As we discussed before, performing zero-padding on input data, improves detection accuracy however it increases: the number of parameters and memory requirement by $9\times$, power consumption by $1.7\times$, latency by $42\times$ and energy by $68\times$.

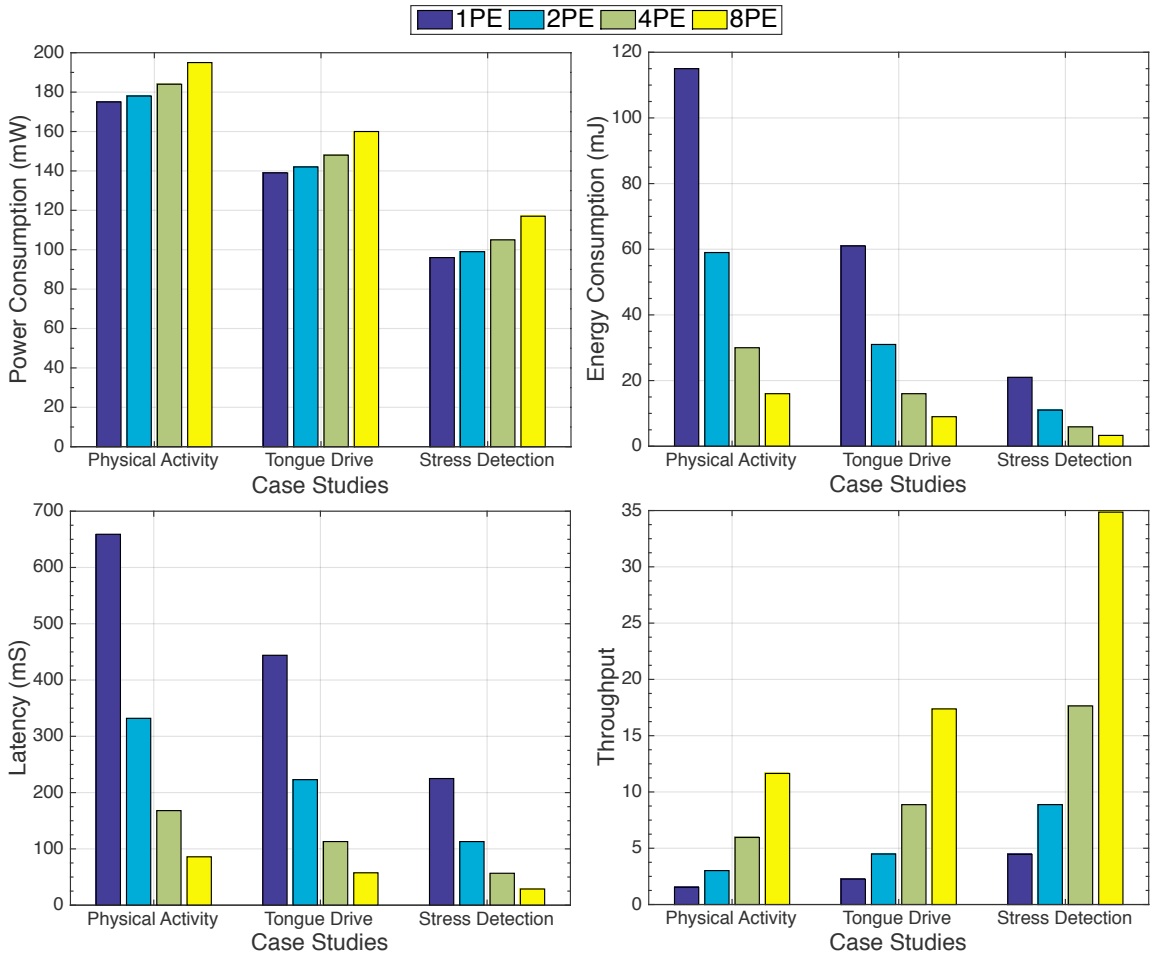

FIG. 6.2. Impact of # PEs on power consumption, energy consumption, latency and throughput for Physical Activity Monitoring, sdTDS and Stress Detection case studies.

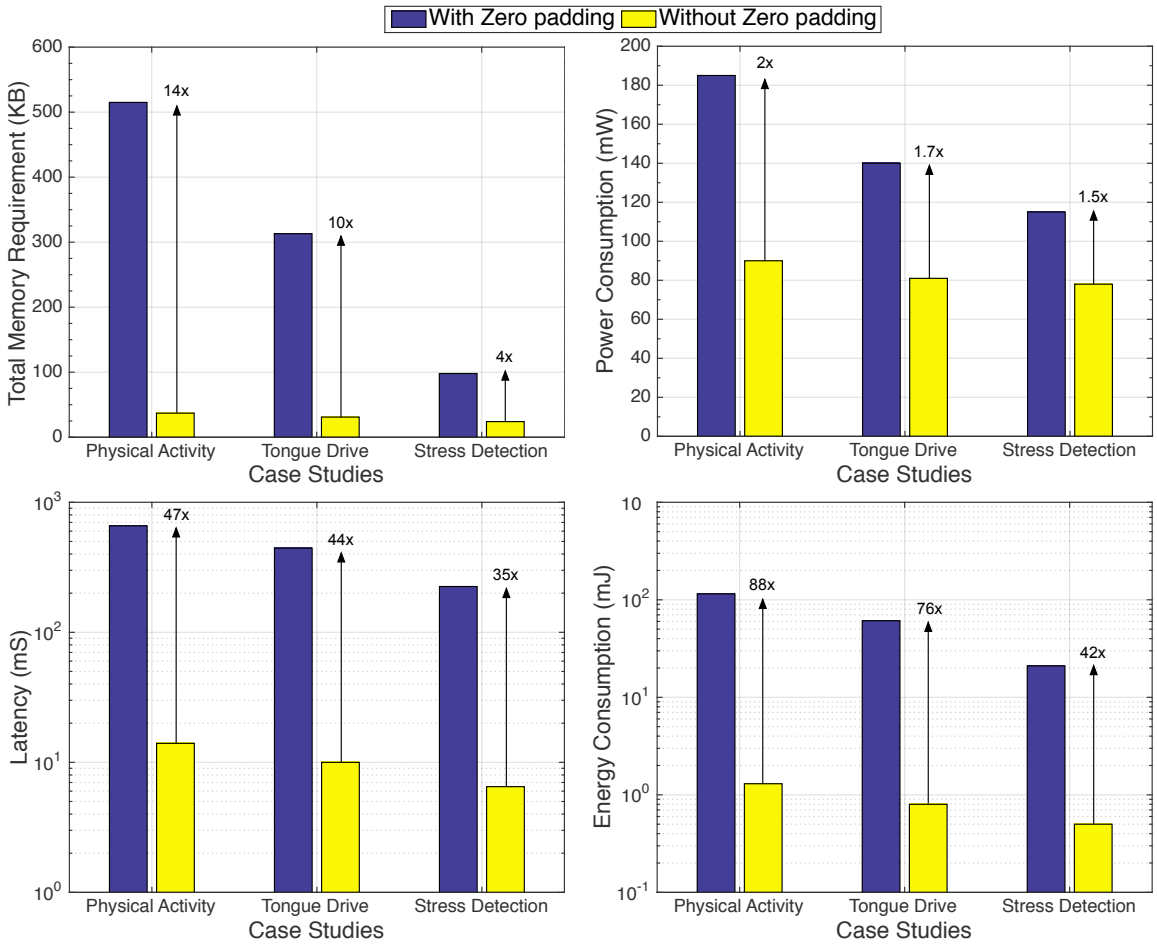

FIG. 6.3. Impact of Zero-padding on FPGA Results.

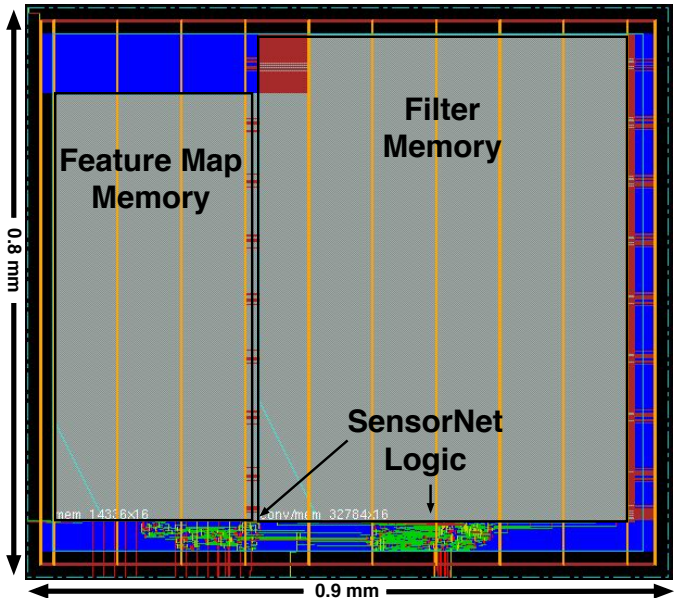

FIG. 6.4. ASIC implementation results at Operating frequency of 50 MHz.

6.2 ASIC Implementation Results and Analysis

To reduce the overall power consumption, 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 layout is shown in Figure 6.4. The results are provided in table 6.3. The SensorNet is able to operate at 540 MHz clock frequency. However, the clock frequency has been reduced to 50 MHz to reduce the power consumption. The ASIC implementation reduces the processor power consumption by a factor of 7.

Table 6.3. SensorNet ASIC implementation results at operating frequency of 50 MHz. CMOS fabrication process is 65 nm with 1 V Power Supply

Metrics	Stress Detection
Design	Fully-serial
Place & route area utilization (%)	90
Operating frequency (MHz)	50
Maximum clock frequency (MHz)	540
Core area (mm^2)	0.7
Latency (mS)	225
Leakage power (mW)	10.3
Dynamic power (mW)	3
Total power (mW)	13.3
Energy (mJ)	3

6.3 Off-the-shelf Targeted Platforms

The proposed SensorNet is built and implemented on the NVIDIA Jetson TX2 platform using the TensorFlow framework. Table 6.4 shows the real-time implementation results in terms of execution time, energy consumption, throughput and energy efficiency, for Physical Activity Monitoring case study. The TX2 base 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 1139 mJ. Also, it takes 699 mS to finish all the necessary computations and the throughput is 1.4 Window/S. (40 channels of data with a window of 64 samples per each channel). For the same input data, when the GPU is enabled and processing is performed on the GPU with a clock frequency of 1.3 GHz and all CPU cores are on, running at 345 MHz, the energy consumption is 25 mJ, the execution time is 7.7 mS and the throughput is 130 Window/S which shows an improvement of 46×, 91× and 93× compared to the TX2 base setting, respectively.

Table 6.4. Real-time SensorNet implementation results on NVIDIA Jetson TX2 SoC (CPU+GPU). The TX2 base refers to using a single CPU core running at the lowest clock freq. of 345 MHz with GPU disabled.

Specifications/Platform	**NVIDIA TX2 (Base)**	**NVIDIA TX2 With GPU**	**Improvement over Base**
Execution time (mS)	699	7.7	91×
Energy consumption (mJ)	1139	25	46×
Throughput (Window/s)	1.4	130	93×
Energy efficiency (Window/s/W)	1.06	50.84	48X

Fig. 6.5 and 6.6 provide insight into the performance of the proposed processor implemented on the NVIDIA Jetson TX2. Each bar represents a different configuration of active CPUs for TX2. Fig. 6.5 provides specific results for CPU only while Fig. 6.6 provides the results of exploiting the GPU.

Fig. 6.5 part (a) is the energy consumption of the proposed MDCNN processor only on the CPU cores. The lowest energy consumption is 96 mJ using all 6 CPU cores clocked at 1113 MHz. Similarly, Fig. 6.6 part (a) shows the energy consumption of using the GPU for feature extraction and classification. The best energy consumption out of all possible configurations is 25 mJ, which is achieved using all CPUs cores configured at a frequency of 345 MHz and the GPU clocked at 1.3 GHz.

Figure. 6.5 part (b) and 6.6 part (b) demonstrate the ability to decrease the execution time when moving from the base single TX2 ARM A57 core to other configurations. The CPU configurations benefit first from increasing the clock speed and then from enabling more cores. When only utilizing the CPU, the best configuration is achieved with increasing the number of active ARM A57 cores and setting the core speed to 2035 MHz providing the minimum execution time of 14 mS. For the GPU, the best possible configuration is to have all CPU cores on, running at a frequency of 345 MHz with the GPU clocked at 1.3 GHz, which provides an execution time of 7.7 mS.

Figure. 6.5 part (c) and 6.6 part (c) compare the throughput of the proposed processor in different configurations. The base configuration has a throughput of approximately 1.4 Window/second. By configuring all CPU cores at a frequency of 2035 MHz with the GPU

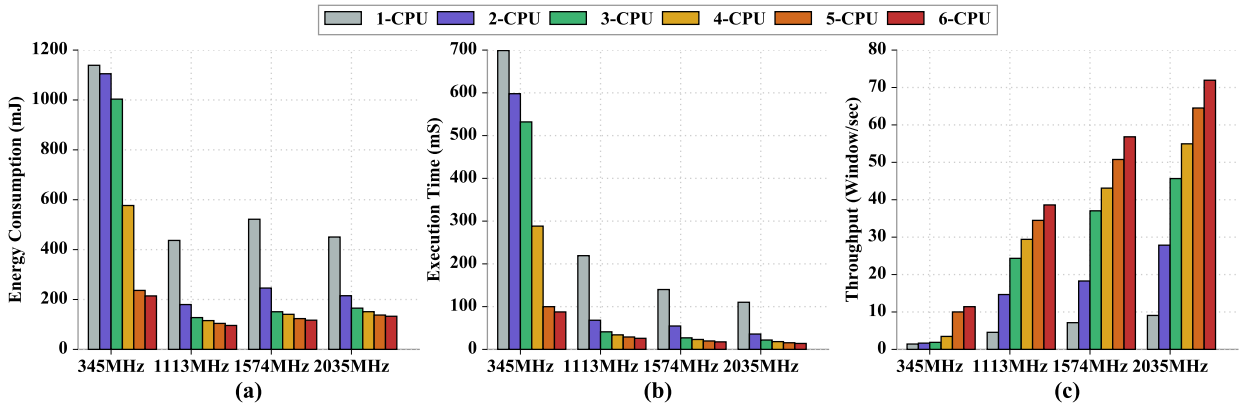

FIG. 6.5. Comparison of the performance of the proposed SensorNet processor in terms of (a) Energy consumption, (b) Execution time, and (c) Throughput when implemented using only ARM Cortex-A57 and Denver 2 CPU cores at different clock frequencies. In this experiment the GPU is not used and therefore disabled.

disabled, the throughput jumps to almost 71 Window/second. The best configuration of the TX2 with GPU enabled is 130 Window/S, with all CPU cores running at 345 MHz and the GPU clocked at 1.3 GHz.

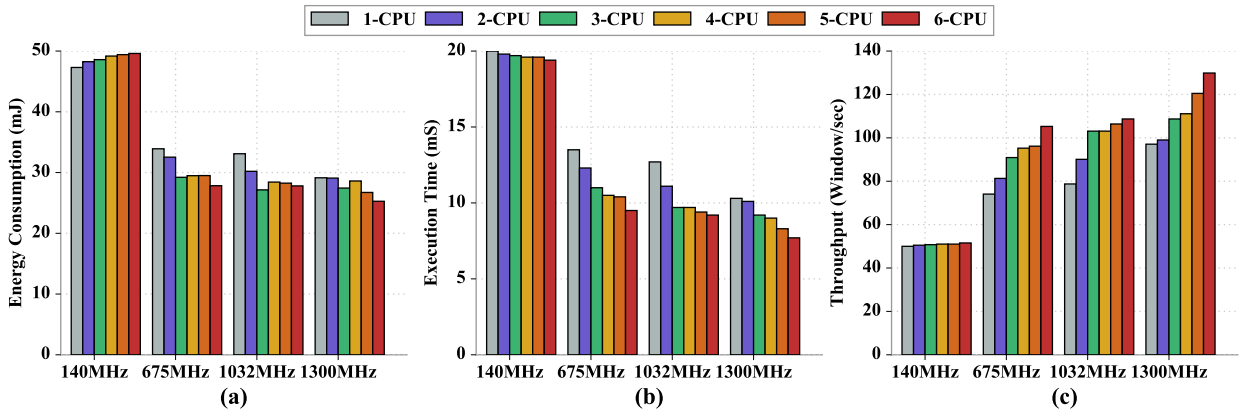

FIG. 6.6. Comparison of the performance of the proposed SensorNet in terms of (a) Energy consumption, (b) Execution time, and (c) Throughput when implemented using the GPU at different clock frequencies. In this experiment, ARM Cortex-A57 and Denver 2 CPU cores are running at lowest frequency of 345 MHz.

6.4 Comparison of SensorNet performance on Different Embedded Platforms

Table 6.5. SensorNet Implementation results on FPGA, CPU and GPU platforms in terms of power consumption, throughput, energy and execution time, for the Physical Activity Monitoring case study.

Platform	**NVIDIA TX2 (Base)**	**NVIDIA TX2 With GPU**	**FPGA Fully-serial**	**FPGA Fully-parallel**
Operating frequency (MHz)	350	140	50	50
Latency (mS)	699	20	659	85
Throughput (Window/s)	1.4	50	1.5	12
Power consumption (mW)	1630	2365	175	196
Energy consumption (mJ)	1139	48	115	16
Energy efficiency (Window/s/W)	1.22	21	13	61

Table 6.5 shows a comparison between our implementation on different embedded platforms including FPGA and NVIDIA TX2 SoC. The TX2 base refers to using a single CPU core running at the lowest clock freq. of 345 MHz. The TX2 with GPU is configured when the processing is performed on the GPU with a clock frequency 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.

6.5 Comparison with Existing Work

Table 6.6 shows a comparison of the proposed SensorNet hardware implementation results with existing multimodal deep learning solutions on embedded devices. When SensorNet is deployed on Xilinx FPGA platform in a full-parallel way and running at 50

MHz, it consumes 3.5 mJ energy which is 200×, 63×, 27× lower compared to the prior proposed embedded implementation [Yao *et al.*2017] and [Radu *et al.*2016], respectively. To have a fair comparison, we provide Stress Detection results as it has similar number of input channels compared to [Yao *et al.*2017] and [Radu *et al.*2016]

Table 6.6. Comparison of the SensorNet performance with related works when evaluated on real-time embedded settings.

Metrics	**[Yao *et al.*2017]**	**[Yao *et al.*2017]**	**[Radu *et al.*2016]**	**This work**
Application	Human activity recognition	Human activity recognition	Activity recognition	Stress detection
# Input channels	6	6	6	7
Technique	DeepSense	DeepSense	Multimodal RBM	SensorNet
Platform	Intel Edison	Nexus 5	Qualcomm Snapdragon	Xilinx FPGA
Latency (mS)	105	38	50	28
Energy consumption (mJ)	700	220	96	3.5

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In this dissertation we proposed contributions in three key enterprises to enable deploying multimodal data classification using deep neural networks in low power and resource-bound embedded settings. In the first enterprise, we proposed the SensorNet which is a scalable deep convolutional neural network designed to classify multimodal time series signals. The raw time series signals generated by different sensor modalities with different sampling rates are first fused into images; then, a Deep Convolutional Neural Network (DCNN) is utilized to automatically learn shared features in the images and perform the classification. Proposing SensorNet has the following advantages: 1) SensorNet is scalable as it can process different types of time series data with variety of input channels and sampling rates. 2) there is no need to employ separate signal processing techniques for processing the data generated by each sensor modality. 3) an expert knowledge for extracting features for each sensor data is not required. 4) is easy

and fast for SensorNet to adapt to new sensor modalities with a different sampling rate. (5) very high detection accuracies for different case studies are achieved. (6) SensorNet has a very efficient architecture which makes it suitable to be deployed in a low-power and resource-limited embedded platform. SensorNet performance is evaluated using three different case studies including Physical Activity Monitoring, sdTDS and Stress Detection and it achieves an average detection accuracy of 98%, 96.2% and 94% for each case study, respectively.

In the second enterprise, we proposed and designed a custom hardware architecture for the efficient deployment of SensorNet on low-power and resource-limited embedded settings such as FPGAs and ASICs, which can perform the entire SensorNet signal processing in real-time. We implemented SensorNet using our custom hardware architecture on Xilinx FPGA and it consumes 176 mW, 140 mW and 96 mW power for Physical Activity Monitoring, sdTDS and Stress Detection case studies, respectively. To further reduce the power consumption, SensorNet is implemented using ASIC at the post layout level in 65-nm CMOS technology which consumes approximately 7$\times$ lower power compared to the FPGA implementation. Additionally, SensorNet is implemented on NVIDIA Jetson TX2 SoC (CPU+GPU) which is an embedded commercial off-the-shelf platform. Compared to TX2 single-core CPU and GPU implementations, FPGA-based SensorNet obtains 8$\times$ and 12$\times$ improvement in power consumption, and 71$\times$ and 3$\times$ improvement in energy consumption. Furthermore, SensorNet achieves 200$\times$, 63$\times$, 27$\times$ lower energy consumption compared to previous related work. In the third enterprise, we proposed

and developed a stand-alone dual-mode Tongue Drive System (sdTDS) which employs SensorNet to perform all required multimodal signal processing in real-time. sdTDS 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 and head movements.

It is worth mentioning that, SensorNet is considered as a generic deep neural network that can accommodates a wide range of applications with minimal effort.

7.2 Future Work

Going forward there exists a number of possible directions to further improve upon all of the three enterprises. For the SensorNet architecture, there is potential to employ Long Short Term Memory (LSTM). LSTMs are a special kind of Recurrent Neural Network(RNN), capable of learning long-term dependencies. LSTMs have shown great success for time series data classifications. Also, LSTMs can be joined with convolutional layers to form a network for multimodal data classification. In terms of SensorNet hardware architecture, one direction is looking to exploit other forms of parallelization. In addition, to parallelizing across output channels SensorNet could also be targeted to parallelize by patches within a channel. Finally for the sdTDS, there is a potential to add more sensor modalities to the headset. Adding a microphone to the headset makes the user more independent.

REFERENCES

[Alam & Hamida2014] Alam, M. M., and Hamida, E. B. 2014. Surveying wearable human assistive technology for life and safety critical applications: Standards, challenges and opportunities. *Sensors* 14(5):9153–9209.

[Barea & others2002] Barea, R., et al. 2002. System for assisted mobility using eye movements based on electrooculography. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 10(4):209–218.

[Batista, Wang, & Keogh2011] Batista, G. E.; Wang, X.; and Keogh, E. J. 2011. A complexity-invariant distance measure for time series. In *SDM*, volume 11, 699–710. SIAM.

[Birjandtalab *et al.*2016] Birjandtalab, J.; Cogan, D.; Pouyan, M. B.; and Nourani, M. 2016. A non-eeg biosignals dataset for assessment and visualization of neurological status. In *Signal Processing Systems (SiPS), 2016 IEEE International Workshop on*, 110–114. IEEE.

[Caltenco *et al.*2012] Caltenco, H. A.; Lontis, E. R.; Boudreau, S. A.; Bentsen, B.; Struijk, J.; and Struijk, L. N. 2012. Tip of the tongue selectivity and motor learning in the palatal area. *Biomedical Engineering, IEEE Transactions on* 59(1):174–182.

[Carlson & Ehrlich2005] Carlson, D., and Ehrlich, N. 2005. *Assistive Technology and*

information technology use and need by persons with disabilities in the United States, 2001. US Department of Education, National Institute on Disability and Rehabilitation Research.

[Chen *et al.*2015] Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; and Batista, G. 2015. The ucr time series classification archive.

[Chollet & others2015] Chollet, F., et al. 2015. Keras.

[Collobert & Weston2008] Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. ACM.

[Cook *et al.*2005] Cook, A. M.; Bentz, B.; Harbottle, N.; Lynch, C.; and Miller, B. 2005. School-based use of a robotic arm system by children with disabilities. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 13(4):452–460.

[Courbariaux & others2015] Courbariaux, M., et al. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, 3123–3131.

[Courbariaux *et al.*2016] Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

[Dauphin & Bengio2013] Dauphin, Y. N., and Bengio, Y. 2013. Big neural networks waste capacity. *arXiv preprint arXiv:1301.3583*.

[Ding *et al.*2008] Ding, H.; Trajcevski, G.; Scheuermann, P.; Wang, X.; and Keogh, E. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1(2):1542–1552.

[Genari *et al.*2013] Genari, C. M.; Bellini, B. S.; Fernandes, P. T.; Gabriela, C.; Lima, F. O.; and Li, L. M. 2013. Perception of bci assistive technology by post-ischemic stroke patients. In *Biosignals and Biorobotics Conference (BRC), 2013 ISSNIP*, 1–5. IEEE.

[Girshick *et al.*2014] Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Gomez, Oller, & Paradells2012] Gomez, C.; Oller, J.; and Paradells, J. 2012. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors* 12(9):11734–11753.

[Gong & others2014] Gong, Y., et al. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

[Graves, Mohamed, & Hinton2013] Graves, A.; Mohamed, A.; and Hinton, G. E. 2013. Speech recognition with deep recurrent neural networks. *CoRR* abs/1303.5778.

[Guan & Ploetz2017] Guan, Y., and Ploetz, T. 2017. Ensembles of deep lstm learners for activity recognition using wearables. *arXiv preprint arXiv:1703.09370*.

[Han, Mao, & Dally2015] Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

[Hinton & others2012] Hinton, G., et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29(6):82–97.

[Horki *et al.*2015] Horki, P.; Klobassa, D. S.; Pokorny, C.; and Muller-Putz, G. R. 2015. Evaluation of healthy eeg responses for spelling through listener-assisted scanning. *Biomedical and Health Informatics, IEEE Journal of* 19(1):29–36.

[Huang & others2006] Huang, C.-N., et al. 2006. Application of facial electromyography in computer mouse access for people with disabilities. *Disability and Rehabilitation* 28(4):231–237.

[Huo & others2007] Huo, X., et al. 2007. Using magneto-inductive sensors to detect tongue position in a wireless assistive technology for people with severe disabilities. In *Sensors, 2007 IEEE*, 732–735. IEEE.

[Huo *et al.*2013] Huo, X.; Park, H.; Kim, J.; and Ghovanloo, M. 2013. A dual-mode human computer interface combining speech and tongue motion for people with severe disabilities. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 21(6):979–991.

[Huo, Wang, & Ghovanloo2008] Huo, X.; Wang, J.; and Ghovanloo, M. 2008. A magneto-inductive sensor based wireless tongue-computer interface. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 16(5):497–504.

[Hwu & others2017] Hwu, T., et al. 2017. A self-driving robot using deep convolutional neural networks on neuromorphic hardware. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, 635–641. IEEE.

[isc]

[Jafari *et al.*2015] Jafari, A.; Page, A.; Sagedy, C.; Smith, E.; and Mohsenin, T. 2015. A low power seizure detection processor based on direct use of compressively-sensed data and employing a deterministic random matrix. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, 1–4. IEEE.

[Jafari *et al.*2017] Jafari, A.; Buswell, N.; Ghovanloo, M.; and Mohsenin, T. 2017. A low power wearable stand-alone tongue drive system for people with severe disabilities. *IEEE Transactions on Biomedical Circuits and Systems*.

[Jafari, Ghovanloo, & Mohsenin2017] Jafari, A.; Ghovanloo, M.; and Mohsenin, T. 2017.

An embedded fpga accelerator for a stand-alone dual-mode assistive device. In *IEEE Biomedical Circuits and Systems (Biocas) Conference*.

[Jiang & Yin2015] Jiang, W., and Yin, Z. 2015. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*, 1307–1310. ACM.

[Kampouraki, Manis, & Nikou] Kampouraki, A.; Manis, G.; and Nikou, C. Heartbeat time series classification with support vector machines. *IEEE Transactions on Information Technology in Biomedicine* 13(4).

[Kandel & others2000] Kandel, E. R., et al. 2000. *Principles of neural science*, volume 4. McGraw-hill New York.

[Kowtko2012] Kowtko, M. 2012. Using assistive technologies to improve lives of older adults and people with disabilities. In *Systems, Applications and Technology Conference (LISAT), 2012 IEEE Long Island*, 1–6. IEEE.

[Krishnamurthy & Ghovanloo2006] Krishnamurthy, G., and Ghovanloo, M. 2006. Tongue drive: A tongue operated magnetic sensor based wireless assistive technology for people with severe disabilities. In *Proceedings of ISCAS*, 4–pp. IEEE.

[Krizhevsky, Sutskever, & Hinton2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

[Lau & OLeary1993] Lau, C., and OLeary, S. 1993. Comparison of computer interface devices for persons with severe physical disabilities. *American Journal of Occupational Therapy* 47(11):1022–1030.

[Leonardis *et al.*2015] Leonardis, D.; Barsotti, M.; Loconsole, C.; Solazzi, M.; Troncossi, M.; Mazzotti, C.; Castelli, V. P.; Procopio, C.; Lamola, G.; Chisari, C.; et al. 2015. An emg-controlled robotic hand exoskeleton for bilateral rehabilitation. *Haptics, IEEE Transactions on* 8(2):140–151.

[Li *et al.*2010] Li, M.; Rozgica, V.; Thatte, G.; Lee, S.; Emken, A.; Annavaram, M.; Mitra, U.; Spruijt-Metz, D.; and Narayanan, S. 2010. Multimodal physical activity recognition by fusing temporal and cepstral information. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18(4):369–380.

[Li *et al.*2017] Li, X.; Zhang, Y.; Zhang, J.; Chen, S.; Marsic, I.; Farneth, R. A.; and Burd, R. S. 2017. Concurrent activity recognition with multimodal cnn-lstm structure. *arXiv preprint arXiv:1702.01638*.

[Lund & others2009] Lund, M. E., et al. 2009. A framework for mouse and keyboard emulation in a tongue control system. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, 815–818. IEEE.

[Matsubara *et al.*2015] Matsubara, M.; Oba, T.; Kadone, H.; Terasawa, H.; Suzuki, K.;

and Iguchi, M. 2015. Wearable auditory biofeedback device for blind and sighted individuals. *IEEE MultiMedia* (1):68–73.

[McFarland & others2008] McFarland, D. J., et al. 2008. Emulation of computer mouse control with a noninvasive brain–computer interface. *Journal of neural engineering* 5(2):101.

[Mekhalfi *et al.*2015] Mekhalfi, M. L.; Melgani, F.; Bazi, Y.; and Alajlan, N. 2015. A compressive sensing approach to describe indoor scenes for blind people. *Circuits and Systems for Video Technology, IEEE Transactions on* 25(7):1246–1257.

[Mimche *et al.*2016] Mimche, S.; Ahn, D.; Kiani, M.; Elahi, H.; Murray, K.; Easley, K.; Sokoloff, A.; and Ghovanloo, M. 2016. Tongue implant for assistive technologies: Test of migration, tissue reactivity and impact on tongue function. *Archives of Oral Biology* 71:1–9.

[Nam & others2012] Nam, Y., et al. 2012. Tongue-rudder: A glossokinetic-potential-based tongue–machine interface. *IEEE Transactions on Biomedical Engineering* 59(1):290–299.

[NS Andreasen Struijk & others2016] NS Andreasen Struijk, L., et al. 2016. Development and functional demonstration of a wireless intraoral inductive tongue computer interface for severely disabled persons. *Disability and Rehabilitation: Assistive Technology* 1–10.

[Nurvitadhi & others2016] Nurvitadhi, E., et al. 2016. Accelerating binarized neural

networks: Comparison of fpga, cpu, gpu, and asic. In *Field-Programmable Technology (FPT), 2016 International Conference on*. IEEE.

[O'Brien & Ruairi2009] O'Brien, A., and Ruairi, R. M. 2009. Survey of assistive technology devices and applications for aging in place. In *Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, 2009. CENTRIC'09. Second International Conference on*, 7–12. IEEE.

[Ordóñez & Roggen2016] Ordóñez, F. J., and Roggen, D. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16(1):115.

[Page & others2017] Page, A., et al. 2017. Sparcnet: A hardware accelerator for efficient deployment of sparse convolutional networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*.

[Pawluk, Adams, & Kitada2015] Pawluk, D. T.; Adams, R. J.; and Kitada, R. 2015. Designing haptic assistive technology for individuals who are blind or visually impaired. *Haptics, IEEE Transactions on* 8(3):258–278.

[Pereira & others2009] Pereira, C. A. M., et al. 2009. Development and evaluation of a head-controlled human-computer interface with mouse-like functions for physically disabled users. *Clinics* 64(10):975–981.

[Radu *et al.*2016] Radu, V.; Lane, N. D.; Bhattacharya, S.; Mascolo, C.; Marina, M. K.;

and Kawsar, F. 2016. Towards multimodal deep learning for activity recognition on mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, 185–188. ACM.

[Rajpurkar *et al.*2017] Rajpurkar, P.; Hannun, A. Y.; Haghpanahi, M.; Bourn, C.; and Ng, A. Y. 2017. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*.

[Rastegari & others2016] Rastegari, M., et al. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 525–542. Springer.

[Reiss & Stricker2011] Reiss, A., and Stricker, D. 2011. Introducing a modular activity monitoring system. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 5621–5624. IEEE.

[Reiss & Stricker2012] Reiss, A., and Stricker, D. 2012. Introducing a new benchmarked dataset for activity monitoring. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, 108–109. IEEE.

[Ross2001] Ross, D. A. 2001. Implementing assistive technology on wearable computers. *Intelligent Systems, IEEE* 16(3):47–53.

[Sadeghian, Huo, & Ghovanloo2011] Sadeghian, E. B.; Huo, X.; and Ghovanloo, M. 2011. Command detection and classification in tongue drive assistive technology.

In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, 5465–5468. IEEE.

[Sahadat & others2015] Sahadat, M., et al. 2015. A multimodal human computer interface combining head movement, speech and tongue motion for people with severe disabilities. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, 1–4. IEEE.

[Saponas & others2009] Saponas, T. S., et al. 2009. Optically sensing tongue gestures for computer input. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, 177–180. ACM.

[Sarji2008] Sarji, D. K. 2008. Handtalk: Assistive technology for the deaf. *Computer* 41(7):84–86.

[Sartori *et al.*2012] Sartori, M.; Reggiani, M.; Pagello, E.; and Lloyd, D. G. 2012. Modeling the human knee for assistive technologies. *Biomedical Engineering, IEEE Transactions on* 59(9):2642–2649.

[Seelye *et al.*2012] Seelye, A. M.; Schmitter-Edgecombe, M.; Das, B.; and Cook, D. J. 2012. Application of cognitive rehabilitation theory to the development of smart prompting technologies. *Biomedical Engineering, IEEE Reviews in* 5:29–44.

[Seto, Zhang, & Zhou2015] Seto, S.; Zhang, W.; and Zhou, Y. 2015. Multivariate time series classification using dynamic time warping template selection for human

activity recognition. In *Computational Intelligence, 2015 IEEE Symposium Series on*, 1399–1406. IEEE.

[Struijk2006] Struijk, L. N. A. 2006. An inductive tongue computer interface for control of computers and assistive devices. *IEEE Transactions on biomedical Engineering* 53(12):2594–2597.

[Umuroglu & others2017] Umuroglu, Y., et al. 2017. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the SIGDA*. ACM.

[Van Erp, Lotte, & Tangermann2012] Van Erp, J. B.; Lotte, F.; and Tangermann, M. 2012. Brain-computer interfaces: beyond medical applications. *Computer* (4):26–34.

[Vepakomma *et al.*2015] Vepakomma, P.; De, D.; Das, S. K.; and Bhansali, S. 2015. A-wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities. In *Wearable and Implantable Body Sensor Networks (BSN), 2015 IEEE 12th International Conference on*, 1–6. IEEE.

[Viseh, Ghovanloo, & Mohsenin2015] Viseh, S.; Ghovanloo, M.; and Mohsenin, T. 2015. Toward an ultralow-power onboard processor for tongue drive system. *Circuits and Systems II: Express Briefs, IEEE Transactions on* 62(2):174–178.

[Wang & Oates2015] Wang, Z., and Oates, T. 2015. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Wolpaw & others2002] Wolpaw, J. R., et al. 2002. Brain–computer interfaces for communication and control. *Clinical neurophysiology* 113(6):767–791.

[Xu *et al.*2015] Xu, K.; Ba, J.; Kiros, R.; Courville, A.; Salakhutdinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

[Yan & Yu2015] Yan, W., and Yu, L. 2015. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*.

[Yao *et al.*2017] Yao, S.; Hu, S.; Zhao, Y.; Zhang, A.; and Abdelzaher, T. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*, 351–360. International World Wide Web Conferences Steering Committee.

[Yousefi *et al.*2012] Yousefi, B.; Huo, X.; Kim, J.; Veledar, E.; and Ghovanloo, M. 2012. Quantitative and comparative assessment of learning in a tongue-operated computer input device—part ii: Navigation tasks. *Information Technology in Biomedicine, IEEE Transactions on* 16(4):633–643.

[Zhang & others2015] Zhang, Z., et al. 2015. Enhancements of a tongue-operated robotic rehabilitation system. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, 1–4. IEEE.

[Zhang *et al.*2015] Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; and Cong, J. 2015. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, 161–170. New York, NY, USA: ACM.

[Zhao & others2017] Zhao, R., et al. 2017. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *FPGA*, 15–24.

[Zheng *et al.*2014] Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; and Zhao, J. L. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, 298–310. Springer.