FPGA-based Edge Inferencing for Fall Detection

Kishore Bharathkumar
Electrical and Computer Engineering
San Diego State University
San Diego, USA
kishorebharath14@gmail.com

Christopher Paolini
Electrical and Computer Engineering
San Diego State University
San Diego, USA
paolini@engineering.sdsu.edu

Mahasweta Sarkar
Electrical and Computer Engineering
San Diego State University
San Diego, USA
msarkar2@sdsu.edu

2020 IEEE Global Humanitarian Technology Conference (GHTC) | 978-1-7281-7388-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/GHTC46280.2020.9342948

***Abstract*—In the geriatric population, physical injuries sustained by an unintentional or an unpredictable fall on a hard surface is the leading cause of injury related morbidity and sometimes mortality. Each year, close to 30% of adults around the age group of 65 fall down at least once. In the year 2015, close to 2.9 million falls were reported, resulting in 33,000 deaths. As much as 61% of elderly nursing home residents fell at some point during their first year of residence.These falls may aggravate the situation leading to bone fracture, concussion, internal bleeding or traumatic brain injury when immediate medical attention is not offered to the person. Delay in course of the event may sometimes lead to death as well. Recently, many studies have come up with wearable devices. These devices that are now commercially available in the market are small, compact, wireless, battery operated and power efficient. This study discusses the findings that the optimal location for a Fall Detection Sensor on the human body is in front of the Shin bone. This is based on the 183 features collected from Inertial Measurement Unit (IMU) sensors placed on 16 human body locations and trained-tested using Convolutional Neural Networks (CNN) machine learning paradigm. The ultimate goal is to develop a mobile, wireless, wearable, low-power medical device that uses a small Lattice iCE40 Field Programmable Gate Array (FPGA) integrated with gyro and accelerometer sensors which detects whether the device wearer has fallen or not. This FPGA is capable of realizing the Neural Network model implemented in it. This Insitu or Edge inferencing wearable device is capable of providing real-time classifications without any Transmitting or Receiving capabilities over a wireless communication channel.**

***Index Terms*—fall detection sensor, FDS, edge inferencing, machine learning, neural networks, FPGA**

I. Introduction

The leading cause of fall is invariably proportional to the age and physical fitness of the person. As a consequence of fall, significant trauma, disability, and dizziness are caused [1]. Falls are next to automobile or motorcycle injuries in terms of economic concern [2]. On an average, close to 36.8 per 100,000 people over the age of 65 in the United States, die only due to fall [3] as shown in the Fig. 1. Studies reveal that nearly 40% of falls require medical attention or lead to restriction of activity or worsening of health conditions. The frequency is higher in the assisted living centers compared to people living in residential homes [4]. Based on these reports, instant aid needs to be offered to prevent a fatal fall situation. Under such circumstances, a Fall Detection Sensor (FDS) or an equivalent device is required to alert the care-giver or an immediate family member [5].

In Low or middle income countries like Thailand, Vietnam, Indonesia, Cambodia, Malaysia and other South-East Asian countries report fall data in limited studies. Thus, the fall percentage is low in the range of 14-34% leading to limited quality in the research pertaining to fall prevention [6]. In sub-Saharan African countries, fall is not at all considered as a risk factor leading to injuries [7].

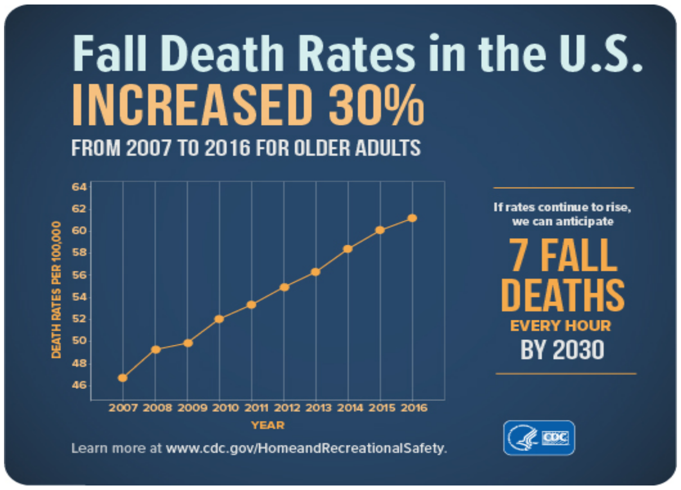

Fig. 1. Fall Deathrates in the US [3]

These Fall Detection Sensors (FDS) are usually small and worn around the neck as a pendant or around the wrist like a watch or carried as mobile phones equipped with accelerometer inside the pant pockets to sense and detect fall events. The makers advertise with a disclaimer stating that "The device may not sense 100% falls. During emergency, press the HELP button". It is highly difficult or impossible for the device wearer to do so when he/she goes into an unconscious state following a fatal fall [8]. Although many types and variants of these Fall detectors are commercially available, it is still not clear as to which is the best or the optimal location on the human body to place a FDS.

In this study, sixteen Inertial Measurement Unit (IMU) Sensors are placed at different locations of the human body which captures up to 183 features. The values recorded are trained and tested using different Machine Learning models to classify an event as a fall or no-fall. Based on these results, it is evident that the Shin bone (below the knee) is the right location to place the sensor on, for maximum accuracy in fall detection. A suitable Neural Network model in Caffe framework, is later implemented on a Lattice iCE40 Ultraplus Breakout board. The board has a FPGA that provides flexible intellectual property (IP) cores that implements Convolutional Neural Networks (CNNs) which computes fall classification in real time, on real time data.

II. Approach and Data set

Experiments on human subjects were carried out in different research labs differently to detect falls. In spite of the fact that all these experiments give details about fall data implementation, the methods followed to record a fall or how it takes place is not mentioned clearly [5]. Although the details of the human subjects involved are disclosed in few research works, the cause for a fall or the measures taken to create a fall is still a mystery [9]. Some studies have used cameras to record videos of fall and use those as input data to train neural network models. Privacy of the subject is compromised in this case.

In our study, the fall experiments were performed at the Neuromechanics and Neuroplasticity Lab of San Diego State University in the Department of Exercise and Nutritional Science (ENS). The lab was setup with Wireless 3D Motion capture cameras which records the human subject movements. As shown in Fig. 2, the Noraxon myoMotion Research Inertial Measurement Unit(IMU) sensors measure the Orientation angles, Anatomical(Joint) Angles and Linear acceleration of the subject's motion [10].

Fig. 2. Noraxon myoMotion IMU Sensors and Receiver [11]

A. Fall Experiment

As stated before, 183 features were recorded based on the IMU's placed at 16 different body locations. The different locations on the upper body consist of: head, upper spine, lower spine, pelvis, upper arms, forearms and hands as shown in Fig. 3. The locations on the lower body consist of the thighs, the shanks and the feet as shown in Fig. 4.

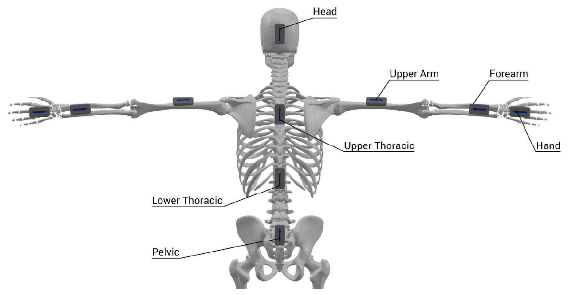

Fig. 3. Upper Body IMU placement [11]

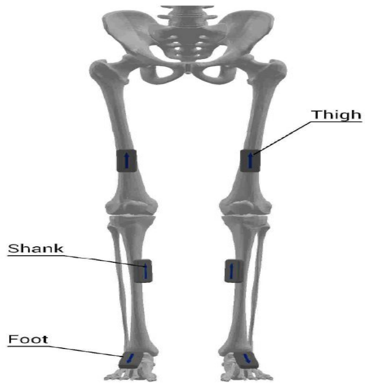

Fig. 4. Lower Body IMU placement [11]

The human subjects involved in fall data collection process were mostly graduate students or faculty members, since since we did not have IRB approval to test on the geriatric population in the first phase of our lab experiments. Upon strapping these sixteen IMU's, information regarding the height, weight, gender and age of the human subject was recorded for calibration purposes. The calibration of the sensors involved stretching of arms, legs, moving back and forth to see if the recordings displayed on the Noraxon MR3 Suite Software were accurately captured on screen. Calibration was an important step in this experiment before every fall trial. Incorrect calibration would impact the measurements of inertial motion extensively.

After carefully calibrating the IMUs, the subject was given a Virtual Reality (VR) headset to ensure that the participant was transported into a virtual world of maybe a walk along a beautiful river bank. Thus the subject was ”removed” from the lab setting and was unaware of when a fall could be ”attempted” on her. The participant was made to walk on a straight path (back and forth) about 7feet long in the lab. The path was covered with a lightweight rug that could be tugged at manually to initiate a fall when the subject was walking on it. The path was surrounded by thick mattresses as precautionary measures to avoid injuries in case of a fall event.

When the subject looked comfortable in the virtual environment (brought about by the VR headset) and seemed to enjoy the ”stroll along the river bed”, a tug at the rug (on which the subject was walking) was initiated manually (by student volunteers) thus inducing a fall or a no fall (stumble or slip). In case of a fall, the subject landed safely on the soft rug or on the mattress surrounding the path which eliminated any harsh impact due to the fall on the subject. After a fall was induced, the sensors were checked for any dislocation and re-calibrated before moving on with the next fall trial. The fall was recorded from the point the subject started walking forward leading to a fall, till the subject was back on his feet following a fall or a stumble.

B. Dataset and Features

The 183 kinematic features determined by the Noraxon IMU’s are displayed on the Noraxon myoRESEARCH Software suite (MR3) which includes the orientation angles, anatomical angles and linear acceleration measured at a sampling rate of 200 Hz. The course, pitch and roll (in degrees) are recorded on the sensors placed on the head, upper thoracic, lower thoracic, pelvic, right and left arms, forearms, hands, thighs, shanks (shin bone) and feet. Also, the acceleration (in milligal or 1 cm s^{-2}) and rotation (in degrees) along X-,Y-, and Z-axis are recorded as well. These sensor measurements are merged together to compute flexion, abduction and rotation of the right and left shoulders and hips. While the flexion is computed for the left and right knees and elbows, the flexion, lateral and axial readings are computed for cevical, lumbar and thoracic regions. The wrist gives the measurements of extension, radial and supination features. These constitute the 183 feature set that was collected and read as a graph and/or tiny wave forms on the Noraxon MR3 software display tool. Fig. 5 showcases such a feature set for one particular individual who participated in our ”fall-experiment”.

These values were stored as Comma-Separated Value (CSV) files which were accessed using the Noraxon MR3 Software or exported to the computer which has Microsoft Excel or similar CSV file readers installed. The Noraxon MR3 software records and displays the changes in orientation angles and acceleration. With the replay feature, each subject’s visual recordings are played back to identify and label that particular recording as a fall or no fall [9]. In each test case, a clean fall is considered as a complete toppling of the individual on the floor/mattresses and the event was labelled with a binary value ’1’. While, merely stumbling or losing balance followed by a quick recovery were considered as ”non-fall” events and were labeled with a ’0’. These CSV files of different human subjects

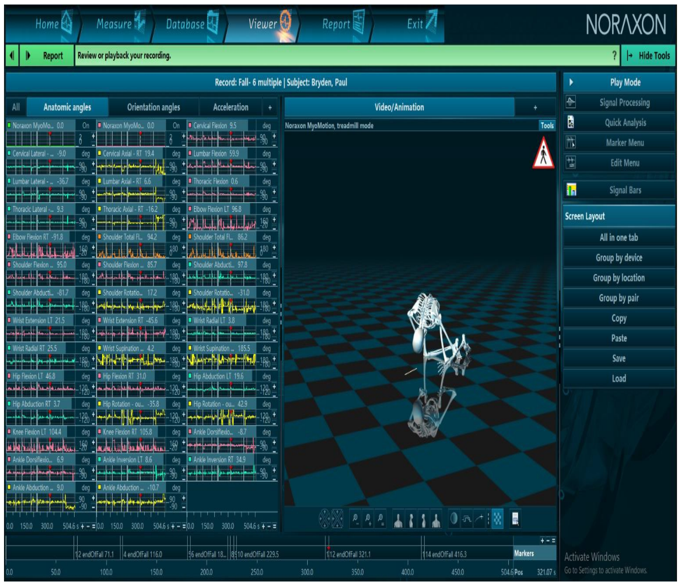

Fig. 5. IMU Measurements on Noraxon Software Suite MR3 [11]

were used as training and testing dataset for four different Machine Learning models discussed in the following section. The 183 feature datasets are available for download at the ’San Diego State University Internet of Things Laboratory’ under the Research section (URL: http://iotlab.sdsu.edu/).

III. Machine Learning Models

Machine learning, as the name suggests, aims to teach the computers how to learn and act without being explicitly programmed. ”Supervised learning” is a widely adopted and the most empirical form of machine learning techniques, which uses a known set of data as input and known responses as output [12]. The model is capable of learning from the observations of known training dataset fed in as input and classifies the test dataset as a fall or non-fall event based on its observations and prior training [13]. Several machine learning approaches exist and can be used to solve a variety of problems. In this research work, we have used four different machine learning techniques on our dataset to obtain the optimal location of strapping a wearable sensor to best detect a fall. For those willing to understand more about the concept of Machine Learning, please refer the book ”Deep Learning (Adaptive Computation and Machine Learning series) by Ian Goodfellow, Yoshua Bengio and Aaron Courville” available in HTML online for free (URL: https://github.com/janishar/mit-deep-learning-book-pdf).

The input data has variable time length segments. The input vectors for walking motion leading to a fall was arbitrarily chosen to exist in the range of 1.5s to 7s with an average time of 3s. On the other hand, in case of the walking motion of non-fall events, the range was chosen to be between 1.5s to

13s with an average time of 7s. In general, the overall mean time for all types of walking was observed to be 5s.

A. Long Short Term Memory-Recurrent Neural Network (LSTM-RNN) Model

The Recurrent Neural Networks have the ability to characterize temporal behavior of a signal as they have loops in their architecture. But these networks experience vanishing gradients problems. LSTM-RNNs comprise of a memory cell to overcome this issue and Gating functions into their state dynamics [14]. A segment of time preceding a fall was trained with RNN using Keras framework running on top of Tensorflow. Tensorflow is created by Google for deep learning purposes and Keras is a higher level API used for BNN, RNN models with faster computations.

The LSTM-RNN model is made of two LSTM layers followed by a Dense layer [15]. While the LSTM layers have dimensions of 183x25 and 25x20, the Dense layer has a dimension of 20x1. The Dense layer does the following operation on the input data:
output = activation(dot(input, kernel) + bias) where, kernel represents the weighted data and dot represents the dot product of all the input data and their corresponding weights [16]. Sigmoid is used as the Activation function in this model.

As stated before, the model was trained on a non-uniform time segment data wherein the time segments comprise of leading up to, and including, a fall. The variable time segment data fed into the model for training goes through one iteration, where one iteration consists of one forward pass and one backward pass of each batch size. The results of LSTM-RNN model after training and testing is plotted as a Receiver Operating Characteristic (ROC) curve for each of the IMU sensors [15].

B. Gradient Boosted Decision Tree (GBT) Model

Gradient Boosting Decision Tree algorithm comprises of multiple iterative decision trees and is an additive algorithm [17]. The results of all the trees are added together and the cumulative value obtained is the final classification result. A collection of GBTs were trained with the recorded fall data using XGBoost shallow learning, scalable tree boosting framework [18]. For example, the final prediction for a given example is the sum of predictions from each tree as represented in the Tree Ensemble model shown in Fig. 6.

The common notion behind GBT model is to split branches, which yields y'[i] a group of predictions, and then compute an error y[i] - y'[i]. GBT as a binary classifier is considered to be the cutting edge model for classification of non-perceptual data like sensor measurements. A logistic function L(x) is used to map the prediction y'[i] to a range [0,1], for a binary classifier which classifies an input data as a fall or no fall. A Mean Squared Error (MSE) function J = $(y[i] - y'[i])^2$ and

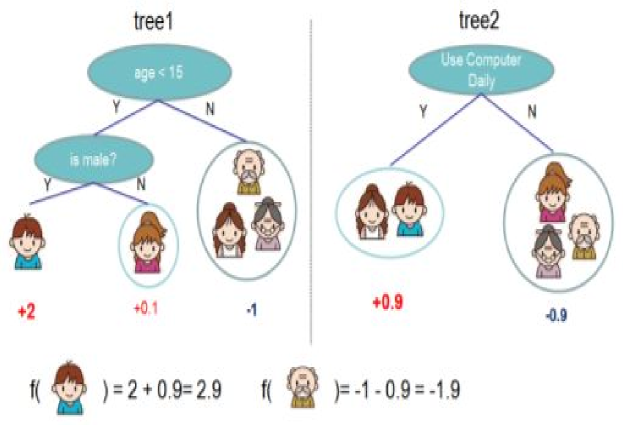

Fig. 6. Gradient Boosted Tree - Example Tree Ensemble Model [18]

a loss function $f[i] = J$ is devised, in order to minimize J where y'[i] = y'[i] - α f[i], for some α. The algorithm is then reiterated for accuracy. The GBT approach, models "loss" as a function of the number of epoch cycles it records [15]. An Epoch is defined as the number of times the entire data set passes through the model.

An average of the ensemble of all trees yields a final classification (Fall or No Fall). Area Under Curve (AUC) under the ROC is chosen as the evaluation metric for validation. The optimal cutoff is the False Positive Rate (FPR) that yields the greatest J = Sensitivity + Specificity - 1 where, J is the Youden Statistic [15].

C. Random Forest Model

Random Decision Forest, comprises of many individual decision trees that operate together as a whole (ensemble learning). The individual class prediction from each of these trees in a random forest is collectively compared and based on the maximum number of votes for a class, the final prediction of the model is made as the result of model's classification [19]. Fig. 7 is an example of how a forest of individual trees outputs a class prediction and the class with the majority value is finalised as the model's output. In this case, there are six 1's and three 0's and the ultimate prediction is 1.

In this approach, from Scikit-Learn's sklearn.ensemble library, RandomForestClassifier( ) model was used [15]. While the Gradient Boosted Tree model reduced Mean Square Error function J (during each iteration) by training the input data sequentially, the Random Forests model performs training in parallel mode. In this approach, we employ a set of 10 independent trees (n_estimators=10) where each tree is trained in parallel and an unified classification result is obtained as an outcome of aggregated solution from the ensemble of trees.

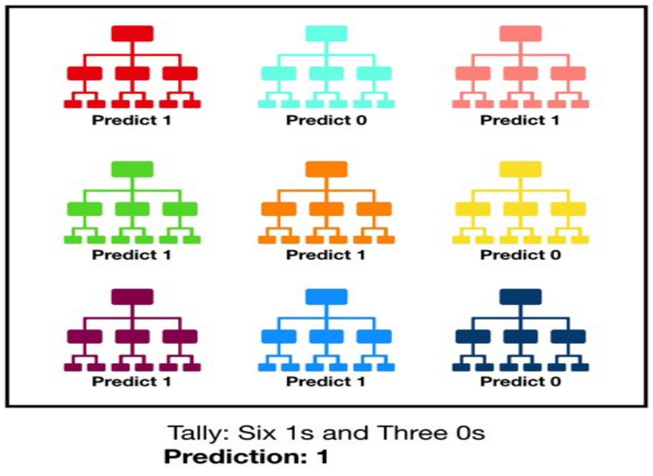

Fig. 7. Visualization of a Random Forest Model making a Prediction [19]

The 'tree' is trained with evaluated Gini impurity indices in Random Forest model. The Gini impurity is seen in every node of the tree which gives the probability of an incorrect classification of a randomly selected test datum from the labelled fall dataset given the selected test datum was either labelled as fall or no-fall randomly [20].

D. Binarized Neural Network Model

The fourth model in this study is a Binarized Neural Network model that functions with binary weights and activations during run-time. These are dedicated to evaluate the parameters gradients [21]. During forward process, both memory size and speed are reduced abruptly compared to the Convolutional Neural Network (CNN) model, where the gradients are not binary during backward propagation. The weights and the activations are constrained to +1 or -1 during run-time [22].

BNN uses Multiply-Accumulate (MAC) operations that are low-level linear algebra operations in deep learning like vector dot product. Because the weights are constrained to +1 and -1, 32-bit floating point MACs are taken over by 1-bit XNOR-popcount operations. A popcount is defined as a binary operation of a 32-bit input word and the output is the word having number of set bits as 1. Thus a standard vector dot product function is compensated by a binraized variant executed by a XNOR-popcount function. This causes the BNN method to provide higher power efficiency during a forward pass. They also demand 32 times fewer memory accesses with a 32 times smaller memory capacity than a 32-bit Deep Neural Network (DNNs) [22]. The BNN model is trained using Keras framework running over Tensorflow. The model comprises of four BinaryDense layers in Keras along with Dropout and Batch Normalization layers [15].

IV. Caffe Model and Implementation

From the above Machine Learning model results, it was observed that the Left Shank sensor gave the best Area Under Curve (AUC) which means that if a sensor is placed at the Shin bone, the accuracy at which a device can predict a fall is much higher than other locations on the human body. Since BNN, Gradient Boosted Trees and Random Forests consider single point in time for training while RNN takes into consideration a segment of time, RNN model outperforms other models and displays higher AUC. Although this is true with regards to the results obtained, in terms of implementation on a hardware, as stated earlier the Binarized Neural Network is much faster than a DNN [22]. Convolutional Neural Network (CNN) is one computationally effective architecture similar to that of the BNN that can be implemented on a Lattice FPGA developed by Lattice Semiconductors.

Considering these factors, small form-factor FPGAs are now commercially available and one such FPGA is found in the Lattice iCE40 Ultra-plus Breakout board [23]. The FPGA supports implementation of Convolutional Neural Network to perform analysis, simulation and as a result, it is able to predict an event as a fall or no-fall in real time. The Caffe model was implemented on a local computer, which consists of eight NVIDIA Tesla V100 PCIe 16GB GPUs. The model is executed on a Supermicro SuperServer SYS-4028GR-TR [24]. A Caffe is actually a Deep Learning framework created by Berkeley AI Research (BAIR) similar to other Machine Learning frameworks. Because of properties like speed and extensible code for active development, Caffe is best suited for research deployment purposes [25].

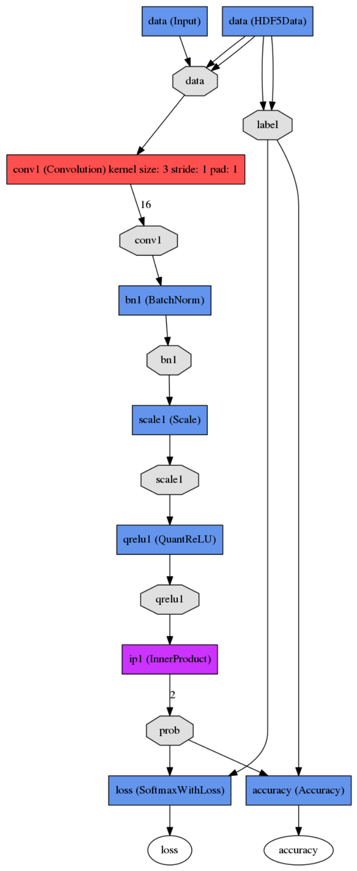

Fig. 8. Convolutional Neural Network model with CBSR layers

A. The Caffe Module

The Convolutional Neural Network Caffe model for fall detection and classification purpose was implemented similar to that of the BNN model in Tensorflow as seen in Fig. 8. While the Caffe model is a hard coded framework, it needs to be written on a text editor and saved as a Proto (.prototxt) file. On top of the existing layers prescribed by BAIR's Caffe framework, Lattice Semiconductors generated an exclusive set of layers for CNN implementations which can be accessed using CaffeLatticeModule [26]. For implementation in Caffe, the input data format accepted is a HDF5 (Hierarchical Data) file format. The HDF5 file format uses these data stored in the form of arrays and is equivalent to a Numpy array [27]. For training-testing purposes using Caffe framework, 3 different architectures compatible with Convolution Neural Network mode of Lattice FPGA is considered.

The first architecture under consideration was the Covolution-BatchNorm-Scaling-ReLU model shortly called as CBSR model. The Dense layers of Keras-Tensorflow are replaced by Inner Product/ Fully connected layer at the end. It mainly calculates the inner product for a binary network. Similarly, Convolutional 2D layer using weights and activation are also used at the top of the Caffe model. The same parameters used in Keras are retained and implemented here. The model also consists of a Batch Normalization layer, which accelerates the training process of the model and is always followed by a scaling layer since droputs are not supported in this framework. The Caffe model Prototxt file along with the the Solver Prototxt file, generates a '.caffemodel' file for every step size interval given, till it reaches the maximum value of iteration. The Solver Prototxt holds the hyper parameters like base learning rate, momentum, iteration cycle, gamma etc. The other two architectures that we considered were fully connected layer architectures with one-dimensional input data (of size 1x9). This is similar to the CBSR model with the binary convolutional layer instead of the standard two-dimensional convolutional layer with two-dimensional input data.

B. Lattice Implementation

It was imperative that we tested our machine learning model on a hardware to understand the time that would be required to compute a decision as ”fall” or ”no fall” based on real time inputs of the sensor that would detect movement. This method of assigning the heavy duty computation to the edge device of the network is called ”Edge Inferencing”. We used an FPGA [28] to test our model. Edge inferencing also means that the sensor recordings are computed at the FPGA device without the need to transmit to a central server, train-test and evaluate there and send the result back to the device via a wireless connection. As shown in Fig. 9, the Lattice iCE40 Ultraplus breakout board FPGA is such a device that offers adaptable or soft intellectual property (IP) device cores that implement Convolutional Neural Networks (CNNs) that can perform continous fall classification and detection at very low power of 1mW (milliWatt) range [23].

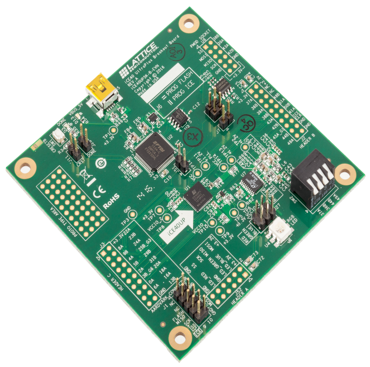

Fig. 9. Lattice iCE40 UltraPlus Breakout Board with FPGA

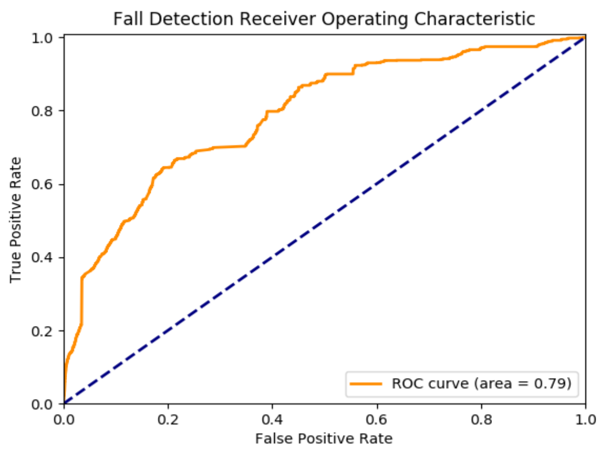

Fig. 10. Receiver Operating Characteristic (ROC) of Convolutional Neural Network model for Fall Detection using Caffe Framework. Highest Area Under Curve (AUC) of 79% for Shank Sensor

In order to implement CNN on this FPGA, Lattice provides a Lattice SenseAI Neural Network Compiler. The compiler is capable of converting the Caffe CNN model into FPGA compatible format by rendering the bitstream (.bin) files which is flashed on to the FPGA. The SenseAI compiler in CNN mode requires the NeuralNet Proto file, the Caffemodel file and a sample input data (two-dimensional array in this case) to analyse, simulate and compile to render the final bitsream file. With the help of Lattice Diamond software, the file is flashed on to the FPGA to perform real time computations based on the sensor input values on the go.

C. Results

The Receiver Operating Characteristic (ROC) curve combined with AUC (Area Under Curve), is used to evaluate a learning algorithm for classification problems. The ROC is obtained as a plot of Sensitivity (True Positive Rate) and

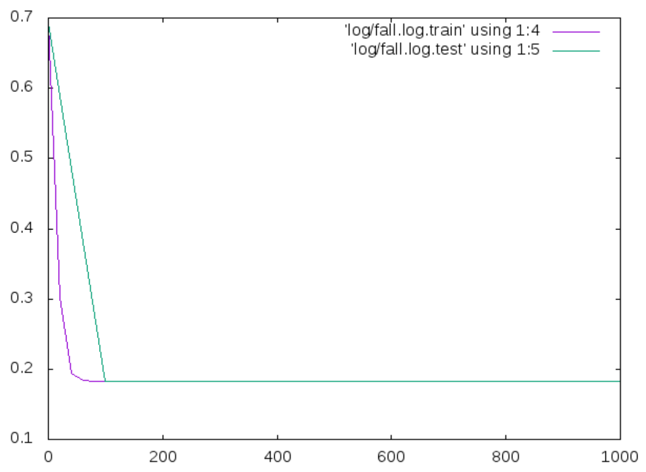

Fig. 11. Loss curve for Shank Sensor as a function of Number of Iterations for Convolutional Neural Network model

1-Specificity (False Positive Rate = 1 - True Negative Rate) is plotted for all the four machine learning models discussed in section III. We have used the Caffe and the Tensorflow framework. For the LSTM-RNN model, the highest AUC is achieved by the Shank sensor (AUC=0.92) placed on the Shinbone, while the left and right wrist locations have an AUC value of 0.91 and 0.86 respectively [15]. The Gradient Boosted Tree model exhibits an AUC of 0.90 for the Shank sensor at the optimal cutoff for FPR (False Positive Rate). Yet again, the Shank sensor on the Left Shin bone offers the highest AUC of 0.85 for the Random Forest model while the Binarized Neural Network (BNN) model also displays an AUC of 0.82 for the Left Shank sensor.

The Convolutional Neural Network designed using Caffe framework was exclusively trained for the left shank sensor readings with 3x3 two-dimensional data as input. The Caffe model generated a Receiver Operating Characteristic (ROC) curve with Area Under Curve (AUC) value of 0.79 as shown in Fig. 10. Also the loss value of 0.18 (training and testing) for the left shank sensor alone as a function of number of iterations when it reaches a minima is shown in Fig. 11.

V. Conclusion

Technological advances are now enabling us to make predictions and classifications in real time in medical devices using in-situ machine learning methods. When a fall occurs, the time it takes to receive attention from a caregiver can worsen the injury resulting in traumatic brain injury, fracture, concussion and bleeding [29]. A leading cause of death among old aged people is because of unforeseen injuries, out of which two-thirds are a result of fall. Since most of the falls take place in assisted living facilities more than residential communities, a Fall Detection System (FDS) should be worn by the elderly persons who are vulnerable to fall.

Datasets on Activities of Daily Life (ADL) and fall events have been published by many Europeon Universities with test case subjects wearing sensors or FDS at different parts of the body like the chest and the thighs, as mobile phones in trouser pockets, as watches on the wrist and as belts along the waist with relatively lesser accuracy in each case. Such devices allow recording of only the linear acceleration, though in reality, the human body has other postures and orientations which are not recorded by these infrastructures. Also, edge computing is not available and uses external memory for computation. From the results, it is evident that all the Machine Learning models in this study exhibit more than 80% classification accuracy with the sensor placed on the shank. The finalised model, that is the Convolutional Neural Network model achieves close to 80% accuracy with Caffe framework and an average of 85% accuracy in Tensorflow framework implemented models. The Tensorflow framework offers higher accuracy when a time series input data is fed into the network. However, the hardware supports the Caffe framework with greater ease and simplicity [30]. The most impactful take-away from our research work is that, future wearable fall detection sensors should be designed to be placed on the human shin bone. Also, these sensors should be designed to such that they are enabled with edge inferencing capabilities so that the computation can be done on-board (for example, on a Lattice like FPGA) with minimal delay and power requirements.

The machine learning models presented in this paper executes the prediction and classification of large amounts of fall data in minimal time (in milliseconds). The FPGA used for the implementation consumes very low power to execute the detection and classification algorithms (in the order of 1 mWatt). The form factor of the FPGA is tiny enough to be embedded in a wearable device. Edge inferencing eliminates the cost of maintaining a cloud infrastructure along with cutting down on the delay involved in transferring data back and forth to the Cloud. The solution that we propose in this paper on a iCE40 FPGA along with the sensors should cost no more than $20-30 when commercially deployed making it a very feasible, low cost and real time solution.

In the future, FDS devices should have a gyroscope and accelerometer to record data and predict whether the patient has fallen. In addition to these features, the FDS can have a GPS (or an indoor trilateration system) inbuilt to inform the care giver with the exact physical location of where the fall has taken place in an assisted living facility [31]. Since RNN has the highest AUC among all the other models because it observes a time segment instead of a single instance of time, implementation on a hardware in real time that supports RNN on a small device with low power requirement needs to be assessed. We plan to extend this project by building our own accelerometer and gyroscope sensors coupled with FPGA on a single device with indoor trilateration system (GPS tracking) and experimenting on more than 100 distinguished human subjects.

Acknowledgement

This work was supported by the HealthLINK Center Cycle 1 Pilot Project *Fall Prediction and Detection in At-Risk Older Adults through Inferencing at the Edge*, sponsored by the National Institute on Minority Health and Health Disparities, National Institutes of Health, and the University Grants Program (UGP) of San Diego State University. The fall experiments were conducted in the "Neuromechanics and Neuroplasticity" laboratory of Dr. Harsimran Baweja in the college of Exercise and Nutritional Sciences (ENS) and SDSU. We are grateful to Dr. Baweja and his graduate student Hedaya for their support [32].

References

[1] C. Hsieh, W. Shi, H. Huang, K. Liu, S. J. Hsu, and C. Chan, "Machine learning-based fall characteristics monitoring system for strategic plan of falls prevention," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, 2018, pp. 818–821.

[2] S. Heinrich, K. Rapp, U. Rissmann, C. Becker, and H.-H. König, "Cost of falls in old age: a systematic review," *Osteoporosis International*, vol. 21, no. 6, pp. 891–902, 2010.

[3] "CDC: Home and Recreational Safety." [Online]. Available: https://www.cdc.gov/homeandrecreationalsafety/falls/fallcost/falls-by-state.html

[4] J. M. Guirguis-Blake, Y. L. Michael, L. A. Perdue, E. L. Coppola, and T. L. Beil, "Interventions to Prevent Falls in Older Adults: Updated Evidence Report and Systematic Review for the US Preventive Services Task Force," *JAMA*, vol. 319, no. 16, pp. 1705–1716, 04 2018.

[5] A. Kurniawan, A. R. Hermawan, and I. K. E. Purnama, "A wearable device for fall detection elderly people using tri dimensional accelerometer," in *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2016, pp. 671–674.

[6] K. D. Hill, P. Suttanon, S.-I. Lin, W. W. Tsang, A. Ashari, T. A. A. Hamid, K. Farrier, and E. Burton, "What works in falls prevention in asia: a systematic review and meta-analysis of randomized controlled trials," *BMC Geriatrics*, vol. 18, no. 1, p. 3, Jan 2018. [Online]. Available: https://doi.org/10.1186/s12877-017-0683-1

[7] M. B. Diamond, S. Dalal, C. Adebamowo, D. Guwatudde, C. Laurence, I. O. Ajayi, F. Bajunirwe, M. A. Njelekela, F. Chiwanga, H.-O. Adami, J. Nankya-Mutyoba, R. Kalyesubula, T. G. Reid, D. Hemenway, and M. D. Holmes, "Prevalence and risk factor for injury in sub-saharan africa: a multicountry study," *Injury Prevention*, vol. 24, no. 4, pp. 272–278, 2018. [Online]. Available: https://injuryprevention.bmj.com/content/24/4/272

[8] "MobileHelp Medical Alert System (2019)." [Online]. Available: https://www.safehome.org/medical-alert-systems/mobilehelp/

[9] A. Harris, H. True, Z. Hu, J. Cho, N. Fell, and M. Sartipi, "Fall recognition using wearable technologies and machine learning algorithms," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 3974–3976.

[10] "Noraxon U.S.A. Inc., myoMOTION Research PRO IMU (2018)." [Online]. Available: https://www.noraxon.com/our-products/research-pro-imu/

[11] "Noraxon U.S.A. Inc., myoMOTION Sensor and Receiver User Manual (2018)." [Online]. Available: https://www.noraxon.com/noraxon-download/myomotion-system-user-manual/

[12] M. V. C. Caya, G. V. Magwili, D. L. Agulto, R. J. Laranang, and L. K. G. Palomo, "Supervised machine learning-based fall detection," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, 2018, pp. 1–6.

[13] H. Yhdego, J. Li, S. Morrison, M. Audette, C. Paolini, M. Sarkar, and H. Okhravi, "Towards musculoskeletal simulation-aware fall injury mitigation: Transfer learning with deep cnn for fall detection," in *2019 Spring Simulation Conference (SpringSim)*, 2019, pp. 1–12.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[15] C. Paolini, D. Soselia, H. Baweja, and M. Sarkar, "Optimal location for fall detection edge inferencing," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec 2019, pp. 1–6.

[16] "Keras-dense layer." [Online]. Available: https://www.tutorialspoint.com/keras/keras{_}dense{_}layer.htm

[17] Y. Ning, S. Zhang, X. Nie, G. Li, and G. Zhao, "Fall detection algorithm based on gradient boosting decision tree," in *2019 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2019, pp. 1–4.

[18] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. Association for Computing Machinery, 2016, p. 785–794.

[19] "Understanding Random Forest." [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees," *Cytometry*, vol. 8, no. 5, pp. 534–535, 1984.

[21] K. Wei, K. Honda, and H. Amano, "Fpga design for autonomous vehicle driving using binarized neural networks," in *2018 International Conference on Field-Programmable Technology (FPT)*, 2018, pp. 425–428.

[22] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.

[23] Lattice Semiconductors, "Accelerating Implementation of Low Power Artificial Intelligence at the Edge," May 2018. [Online]. Available: http://www.latticesemi.com/view_document?document_id=52384

[24] NVIDIA Corporation, "Nvidia® tesla® v100 gpu accelerator," 2018. [Online]. Available: https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[26] Lattice Semiconductors, "Neural Network Compiler BNN Scripts," May 2018. [Online]. Available: https://www.latticesemi.com/-/media/LatticeSemi/Documents/UserManuals/MQ2/FPGA-UG-02055-1-0-Neural-Network-Compiler-BNN-Scripts.ashx?document_id=52387

[27] "HDF5 Input Data format for Caffe." [Online]. Available: http://docs.h5py.org/en/stable/quick.html

[28] S. Dey and A. Mukherjee, "Implementing deep learning and inferencing on fog and edge computing systems," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2018, pp. 818–823.

[29] N. Noury, A. Fleury, P. Rumeau, A. K. Bourke, G. O. Laighin, V. Rialle, and J. E. Lundy, "Fall detection - principles and methods," in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2007, pp. 1663–1666.

[30] S. Satyavolu and A. Bagubali, "Implementation of tensorflow and caffe frameworks: in view of application," in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 2019, pp. 1–4.

[31] K. Gala, P. Bryden, C. Paolini, M. Wang, A. Mihovska, and M. Sarkar, "Real-time indoor geolocation tracking for assisted healthcare facilities," *International Journal of Interdisciplinary Telecommunications and Networking (IJITN)*, vol. 12, pp. 1–21, 04 2020.

[32] "Neuromechanics and neuroplasticity laboratory, sdsu." [Online]. Available: https://ens.sdsu.edu/dpt/research/faculty-research-interests/neuromechanics-and-neuroplasticity-lab/