An Energy-Efficient Low Power
LSTM Processor for Human Activity Monitoring

Arnab Neelim Mazumder, Hasib-Al Rashid, and Tinoosh Mohsenin
Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County
Email: (arnabm1, hrashid1, tinoosh)@umbc.edu

Abstract—A low complexity Long Short-Term Memory (LSTM) based neural network architecture is proposed in this paper for the classification task of recognizing different human activities in relation to various sensor modalities. The proposed model consists of one LSTM layer of 8 units, two dense layers having 80 and 32 neurons respectively and one output layer with 13 neurons for multi-class classification. We achieved 87.17% classification accuracy with our proposed model to classify 12 activities from each other. The proposed work involves extensive hyperparameter optimization in order to develop a hardware implementable model architecture while also maintaining high classification accuracy. In this case, quantization allowed the model to have a small size of 365 kB which resulted in 2x improvement over the 16-bit precision. The hardware architecture is designed in a parameterized way with respect to the number of input channels, filters, and data width to give more flexibility in terms of reconfigurability. The proposed LSTM based model is fully synthesized and placed-and-routed on Xilinx Artix–7 FPGA. Our reconfigurable hardware architecture consumes 82 mW power at an operating frequency of 160 MHz. Our LSTM based FPGA hardware achieves 7.7 GOP/s/W energy efficiency which outperforms previous hardware architecture implementations on Human Activity Recognition (HAR) by atleast 5.2×. The proposed low power LSTM processor also has an improvement of atleast 4.1× for energy efficiency over previous LSTM works based on language modeling and artifact detection.

Keywords: Human Activity Recognition, Physiological Monitoring, Long Short Term Memory (LSTM), FPGA

I. Introduction

Time series classification has been at the forefront of the modern-day research paradigm due to the vast amount of application-specific opportunities that are entwined in our day to day lifestyle. Assessment of time series data has seen a wide range of applications over the years. This has accelerated areas of further research particularly for multimodal time-series data. Nevertheless, the size, nature and dimensionality of these data make the analysis of these signals ever more challenging.

A stream of data collected and sampled at specific time intervals denoting some distinct action is generally called a time series data. Depending on the process of data collection time series data can be univariate or multivariate. Multivariate time series data relates to the multimodal signals which are used widely in different applications. These multimodal signals are captured using accelerometers, magnetometers, gyroscopes, and heart rate monitors sampled at different frequencies. Human Activity Recognition (HAR) is one such application where different algorithmic and application-based procedures are being introduced daily to understand human behavior at its most granular level. Usually, time series problems involving physiological monitoring have been solved using various approaches such as Dynamic Time Warping (DTW) [1], K-nearest neighbors (KNNs) [2], End to End Convolutional Neural Networks (End to End CNN) [3], [4] and Deep Neural Networks (DNNs). However, classification tasks employing KNNs and DTW are associated with long execution time which is not warranted. Recently, DNNs have become very popular for multimodal signal processing [5], [6]. This work introduces an energy-efficient, scalable hardware implementation of the proposed LSTM model illustrated in Fig. 1 that can classify different human activity from multi-channel time-series data that meets the 1-second deadline of data processing time. The major contributions of this paper are as follows:

• Propose an LSTM based physical activity recognition algorithm for multimodal time-series signals.
• Perform substantial hyperparameter tuning with the goal of reducing computation complexity and memory requirements to meet the required accuracy.
• Present an 8-bit quantized LSTM hardware to improve power consumption and memory requirements.
• Implement a parameterized hardware architecture that replicates the algorithm for low-power deployment at embedded application level.
• A comprehensive comparison of the proposed work with state-of-the-art FPGA implementation results.

II. Related Work

There is a plethora of works and algorithmic models that deal with real time-series signals. With the introduction of deep neural networks complex time series data with multiple modalities have been put to use for careful implementation of monitoring and diagnosis scenarios in medical environments [7], [8]. In terms of human activity monitoring, Convolutional Neural Networks (CNN) based models have been broadly used for classification and detection tasks. Convolutional Neural Networks extract spatial features of an image to achieve its desired target. This aspect of the CNNs has been utilized over the years for time series signal assessment. Windowed images at different
Fig. 1: Proposed LSTM based network architecture for human activity monitoring which consists of 1 LSTM layer, 2 dense layers and 1 output (softmax) layer to classify 12 different activities from each other.

samples of real-time signals serve as input to the models which helps the model to figure out the spatial information associated with the data [9]. However, signals associated with physical activity contain concentrated regions of fluctuation or sequence when activity occurs and the windowing process for the CNN architecture takes into account of these sequences during training. This sequential nature of physiological signals allows recurrent neural networks (RNNs) to be particularly useful in this regard. RNNs have the ability to capture temporal information linked with the multimodal signals to create generalized models. This leads to the use of LSTM networks for human activity recognition. In [10] LSTMs were used to concatenate positive time direction (forward state) and negative time direction (backward state) while also granting provisions for residual connections between stacked cells to alleviate the vanishing gradient issue. This improved both the temporal and spatial dimension recognition rate. The characteristic of LSTMs to avoid long dependencies gives them the edge over basic RNN architectures for any sort of classification task and thus in this work, an LSTM based model architecture has been proposed. Most of these software implementations are not suitable for low power hardware deployment and even though there are LSTM accelerators for domain specific applications, to our knowledge there is no implementation of an energy-efficient LSTM hardware model for physical activity monitoring. The only justification we found for FPGA implementation concerning HAR wearable devices comes from [11] which addresses the adaptation issue of wearable HAR systems for low cost FPGAs.

III. Background Overview

A. Long Short Term Memory Networks (LSTM)

Long Short Term Memory (LSTM) network is the subset of the basic Recurrent Neural Network (RNN) architecture. Traditional RNNs can not process large sequences when tanh or ReLU activation function is used. To overcome this issue RNN networks have been evolved to have two different variations i.e. GRU (Gated Recurrent Unit) and LSTM. The main distinction between these two networks lies in the number of gates. GRU has two gates namely reset and update gates while LSTM networks consist of three gates i.e. input, output, and forget gate. This structural difference allows GRU networks to have fewer parameters, computations, and memory. However, with the increased number of gates LSTM networks have more control over accuracy. Besides, both these variants of the RNN network solve the vanishing gradient problem of basic RNN architectures. The structure of an LSTM unit is showed in Fig. 2. The input gate in the LSTM unit determines which memory content to add while forget gate figures out the previous cell state that is required for future inference. Though both these gates act in conjunction with each other, they are completely independent in terms of information correlation. Additionally, sigmoid and tanh activation functions are commonly used in LSTM networks.

B. Physical Activity Monitoring Dataset (PAMAP2)

The HAR dataset used for implementation is called PAMAP2 [12]. The PAMAP2 dataset contains 9 subjects with 12 different activities. The activities, for example, are lying, sitting, walking, running, etc. There are essentially 13 labels, one for each activity equaling the 12 activities in this case and the other label indicates the transient period between activities. The time-series signals were generated using three IMU (Inertial Measurement Units) sensors (gyroscope, accelerometer, magnetometer) and one heart rate monitor with a sampling frequency of 100 Hz.
(IMU) and 9 Hz (Heart Rate Monitor) respectively. There are 52 channels but only 40 of these channels are valid according to [13]. Besides, only the first 8 subjects have significant data to perform classification and as a result subject 9 is not used during model optimization.

IV. Physical Activity Monitoring using LSTM

The model architecture takes raw time-series signals as input. But as part of preprocessing the raw signals are processed to be window images before they are fed to the model. The architecture determines correlations between sensor modalities using these window images of the raw data. Fig. 1 shows a high-level block diagram of the proposed system illustrating the deep neural network module for LSTM, and Fully Connected (FC) layers.

A. Signal Preprocessing

Raw time-series signals consist of $F$ features with the same or different sampling frequency. To generate an image from the variables, a sliding window of size $W$ and increment-step $I$ is passed through all variables, creating a set of images of shape $1 \times W \times F$ (single channel image). The label associated with this image depends on the dataset. 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 $t$ has the prior states of each variable from $(t-W+1)\ldots t$. Thus, the network can look back $W$ prior states of each variable and given the current state of each variable, predicts the label.

B. Neural Network Architecture

The LSTM layer consists of 8 neurons and has a timestep of 64 for this case study. The other part of the input to the LSTM block is the feature which is the number of multimodal channels which is 40 for the PAMAP2 dataset. Three FC layers are utilized in this model architecture with the first and second ones containing 80 and 32 neurons, respectively. The last layer has a size equivalent to the class labels with a SoftMax activation. Tanh activation is used in the LSTM layer with a recurrent activation of hard sigmoid to counter the vanishing gradient issue and to memorize the cell states. Furthermore, FC layers are applied with ReLU activation logic which conforms to the linearity property for values greater than zero and outputs a zero for negative values. The network is optimized using the optimizer Adam which adapts the learning rate in relation to the gradient descent. Also, Categorical cross-entropy is used as the loss function. The rationale behind the number of neurons, LSTM layers, epochs and timesteps will be explained in detail in the next section.

V. Experimental Results and Model Optimization

For the experiment, the dataset for each subject was split to have 80% training data, 10% testing, and 10% validation data. In this section, the results for the experiments in relation to hyperparameter optimization have been discussed to reduce memory requirements while achieving high classification accuracy. The impact of changing the following parameters have been explored here: 1) Number of Epochs, 2) Number of filters in the LSTM layer, and 3) Choice of Timestep.

A. Number of Epochs

The first step towards optimization involved figuring out the number of epochs required to train the model. In this case, we used a preliminary model without unoptimized parameters to see the trend for loss and accuracy at different epochs. The model was trained for 150 epochs while monitoring the validation results for accuracy and loss to find out the desired epoch value. According to this hypothesis, Fig. 3 shows that accuracy becomes stable after 100 epochs. Therefore, we decided to tune all our parameters for 100 epochs.

B. Number of filters in the LSTM layer

The number of filters for the LSTM layer is an important parameter that needs further attention. This is significant in the sense that the number of weights generated, and the amount of computations involved in the model is largely dependent on the choice of this parameter. Besides, the number of weights and computations directly affects memory and energy consumption, respectively. In this case,
the model was tested with several filter configurations. The higher the number of filters the better was the accuracy. However, with large filter sizes, the computations and parameters also increase which is not suitable for low power hardware implementation. Hence, an ideal filter size that presents good classification accuracy with a lower number of computations is required. From Fig. 4 it is evident that the choice of 8 filters provides the best trade-off (46k parameters, 0.52 million computations, 88.84% accuracy) for the memory and computation constraints involved.

C. Choice of Timestep

In deep neural networks that have LSTM modules, the temporal information hinges on the timestep. Timestep acts as the number of sequences the LSTM block looks back to decide for future prediction. Hence, the timestep is a particularly important parameter that determines the structure of the whole model. The choice of timestep varies from application to application and for human activity recognition problems, the timestep should be selected such that the relevant sequence of time-series information for each activity is available within the chosen timestep length. As mentioned in the previous sections, the sampling frequency for the sensors of this dataset was 100 Hz which means there are 100 samples in 1 second of data. Following this, if a timestep of size 128 is selected there will be overlap between sequences of information for corresponding classes which is not desired and with a timestep size of 32, the sequence will be too small for the LSTM model to make an accurate enough prediction. Thus, a timestep size of 64 for is chosen for this case study which provides the best accuracy given the constraints as evident by Fig. 5.

D. Model Weights Quantization

Reducing model size by applying quantization of the model weights is a popular method. Quantization reduces the complexity of the model and reuses the cache in a better and efficient way with the lower precision weights. Quantization is also power efficient as low precision data movement is more efficient than the higher precision data [14]. Therefore, 8-bit quantization was applied with the help of tensorflow-lite post-training quantization. This resulted in an accuracy of 87.2% which represents a 1.7% drop than the original 16-bit model. The main task of quantizing a tensor is to calculate the two parameters scalar and displacement for value range mapping, by solving the set of equations:

\[
\text{float}_\text{min} = \frac{(\text{fixed}_\text{min} - \text{displacement})}{\text{scaler}}. \tag{1}
\]

\[
\text{float}_\text{max} = \frac{(\text{fixed}_\text{max} - \text{displacement})}{\text{scaler}}. \tag{2}
\]

The \text{fixed}_\text{min} / \text{fixed}_\text{max} are known from the target data type nature, but \text{float}_\text{min} / \text{float}_\text{max} are still needed to solve for scalar and displacement. The trained model only includes values in those weight tensors after a regular training, i.e. it only has min / max data for those tensors, so it can only be post-training weight quantized. Here floating Point weights are in 0.0 - 1.0 range and they are quantized to integer 8, so 0.0 becomes -128 and 1.0 becomes 127. However, from the ?? it can be seen that compared to 16 bit precision model, 8-bit precision model has less memory requirement.

VI. Hardware Architecture Design

Fig. 6 depicts the hardware architecture of our LSTM based human activity recognition algorithm. The main components of the hardware architecture includes the LSTM block, dense layer (FC), memories and the top module state machine. The Keras platform was utilized to perform offline training to obtain the trained weights. The floating-point arithmetic is complex and costly in terms of area, time and power in hardware and Keras by default gives weights in floating-point format. Therefore, the weights were converted into 8-bit fixed-point format. The input data for
the hardware model was also converted into 8-bit fixed point format. The **LSTM block** performs the LSTM layer operation. The computation needed in the LSTM layer is shown in [15] which is matrix-vector multiplication. For reducing the memory storage of weights, the kernel weights and recurrent kernel weights were concatenated into two different memories. Equations were merged in a way that it requires less computational block. As shown in Fig. 6, the LSTM block consists of two multiplication and accumulation (MAC) modules, one accumulator, few registers, multipliers, adders and four memories to store kernel weights, recurrent kernel weights, biases, and cell state memory. **Hard Tanh** and **Hard Sigmoid** activation functions are used dynamically inside this block in order to reduce the hardware complexity defined by the following equations:

$$\text{hardsigmoid}(x) = \begin{cases} 
-1, & x < -1 \\
0.2 \times x + 0.5, & -2.5 \leq x \leq 2.5 \\
1, & x > 2.5 
\end{cases} \quad (4)$$

The output of this block is stored into **LSTM Output Memory**.

The second component **Dense Layer** block performs fully-connected operations. FC layers connect every neuron in input to every neuron in output and it corresponds to one matrix-vector multiplication followed by a bias offset and an activation function. This block includes a MAC (multiply - accumulator) engine, a dynamic sorting logic for SoftMax and ReLU activation function. ReLU activation function was used in the first and second dense layer, and SoftMax activation in the output dense layer for the final classification task. All weights were stored in the weight memory. At the completion of the whole computation for the dense layers, the result was stored in **Output Memory** which overwrote the immediate data from previous dense layer.

**TABLE I: Comparison of 16-bit precision and 8-bit precision hardware implementation**

<table>
<thead>
<tr>
<th>Data Precision/ Performance metrics</th>
<th>16 bit</th>
<th>8 bit</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform (MHz)</td>
<td>22.2</td>
<td>9.5</td>
<td>-3.3x</td>
</tr>
<tr>
<td>Accuracy (%)</td>
<td>88.9</td>
<td>87.2</td>
<td>-1.7%</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>66.6</td>
<td>100.0</td>
<td>1.5x</td>
</tr>
<tr>
<td>latency (ms)</td>
<td>2.4</td>
<td>1.8</td>
<td>-20%</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>114.4</td>
<td>82.3</td>
<td>-25%</td>
</tr>
<tr>
<td>Energy (mJ)</td>
<td>207.4</td>
<td>67.2</td>
<td>-64%</td>
</tr>
<tr>
<td>Energy Efficiency (GOPS/W)</td>
<td>2.5</td>
<td>3.3</td>
<td>-33%</td>
</tr>
</tbody>
</table>

**TABLE II: Comparison of this work with previous work related to Human Activity Recognition (HAR) implemented on FPGA**

<table>
<thead>
<tr>
<th>Application</th>
<th>Platform</th>
<th>Human Activity Recognition</th>
<th>Human Activity Recognition</th>
<th>Human Activity Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LSTM</td>
<td>ATTX7 100t</td>
<td>ATTX7 100t</td>
<td>ATTX7 100t</td>
</tr>
<tr>
<td></td>
<td>Accuracy (%)</td>
<td>90.0</td>
<td>85.0</td>
<td>87.2</td>
</tr>
<tr>
<td></td>
<td>Frequency (MHz)</td>
<td>100.0</td>
<td>150.0</td>
<td>160.0</td>
</tr>
<tr>
<td></td>
<td>Latency (ms)</td>
<td>14.8</td>
<td>25.7</td>
<td>10.4</td>
</tr>
<tr>
<td></td>
<td>Throughput (label/s)</td>
<td>87.5</td>
<td>28.3</td>
<td>1220</td>
</tr>
<tr>
<td></td>
<td>Power (mW)</td>
<td>116.0</td>
<td>30,000</td>
<td>82.0</td>
</tr>
<tr>
<td></td>
<td>Energy (mJ)</td>
<td>1.7</td>
<td>1290</td>
<td>87.2</td>
</tr>
<tr>
<td></td>
<td>Energy Efficiency (GOPS/W)</td>
<td>0.4</td>
<td>1.5</td>
<td>7.7</td>
</tr>
</tbody>
</table>
TABLE III: Comparison of this work with previous work related to LSTM models implemented on FPGA

<table>
<thead>
<tr>
<th>Application</th>
<th>[17]-DeepRNN</th>
<th>[18]</th>
<th>[17]-DeepStore</th>
<th>[19]</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Language</td>
<td>Language</td>
<td>Language</td>
<td>EEE Artifact</td>
<td>Human Activity</td>
</tr>
<tr>
<td></td>
<td>Modeling</td>
<td>Modeling</td>
<td>Modeling</td>
<td>Detection</td>
<td>Recognition</td>
</tr>
<tr>
<td>Latency (ms)</td>
<td>Xilinx Zynq</td>
<td>Xilinx Zynq</td>
<td>Xilinx Zynq</td>
<td>Artix7 100</td>
<td>Artix7 100</td>
</tr>
<tr>
<td>Throughput (ms)</td>
<td>XC7Z020</td>
<td>XC7Z030</td>
<td>XC7Z02030</td>
<td>1.2</td>
<td>0.8</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>142</td>
<td>100</td>
<td>142</td>
<td>52.6</td>
<td>1220</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>1800</td>
<td>1190</td>
<td>2300</td>
<td>109</td>
<td>82</td>
</tr>
<tr>
<td>Energy Efficiency (GOP/s/W)</td>
<td>0.4</td>
<td>1.9</td>
<td>0.5</td>
<td>0.5</td>
<td>7.7</td>
</tr>
</tbody>
</table>

in [17] named DeepStore is implemented for language processing and utilizes the on-chip memory of the device to achieve low memory bandwidth whereas [18] introduces an RTL (Register Transfer Level) design for LSTM with the goal of low power implementation and throughput optimization. [19] proposed a hardware implementation of LSTM on programmable logic Artix-7 FPGA to detect EEG artifacts. This work edges the previous works in terms of energy efficiency significantly with an improvement of 19.2×, 4.1×, 16.7× and 15.3× over [17]-DeepRNN, [17]-DeepStore, [18] and [19] respectively. Besides, this implementation has an improvement of 1.5× for both latency and throughput when compared to [19]. Along with this, the power consumption of 82 mW for our work is a considerable improvement over all these designs.

VIII. Conclusion

This work proposes and implements an RTL design for LSTM model architecture targeting physical activity monitoring with a classification deadline of one second. The model set forth for classification attains a classification accuracy of 87.2% with tuned hyperparameters for computation and memory optimization. Furthermore, the Verilog RTL demonstrates considerable improvement when compared to recent LSTM based hardware architectures. This is followed by the figure of merit in terms of energy efficiency which provides evidence that this work is at least 4.1× more efficient than contemporary hardware RTL designs. In addition to these, the future goals will involve introducing ASIC performance results for the same architecture along with NVIDIA Jetson TX2 implementation of the model for energy efficiency on a GPU platform.

REFERENCES