2016 INTERNATIONAL CONFERENCE ON INDOOR POSITIONING AND INDOOR NAVIGATION (IPIN), 4-7 OCTOBER 2016, ALCALÁ DE HENARES, MADRID, SPAIN

Binarized-BLSTM-RNN based Human Activity Recognition

Marcus Edel* and Enrico Köppe**
*Freie Universität Berlin/Mathematics and Computer Science, Berlin, Germany. Email: marcus.edel@fu-berlin.de
**BAM Federal Institute for Material Research and Testing, Berlin, Germany. Email: enrico.koeppe@bam.de

***Abstract*—High computational complexity hinders the widespread usage of neural networks, especially in mobile devices, which are often the basis of fine-grained localization technology for ubiquitous health monitoring, context awareness, and indoor location tracking. In this paper, we present a binarized recurrent neural network whose weight parameters, input, and intermediate hidden layer output signals, are all binary-valued, and require only basic bit logic for the evaluation and training process. The proposed Binarized Long Short-Term Memory Network (B-BLSTM-RNN) is especially suitable for resource-constrained environments since it replaces either floating or fixed-point arithmetic with significantly more efficient bitwise operations. The model is based on a bidirectional Long Short-Term Memory Recurrent Neural Network (BLSTM-RNN). Designed to take contextual information into account, the network can process data gathered from different positions, resulting in a system, that's invariant to transformations and distortions of the input patterns. During the forward pass, the B-BLSTM drastically reduce memory size and accesses, and replace most arithmetic operations with bit-wise operations, which is expected to substantially improve power-efficiency. The binarized network is simple, accurate, efficient, and works on challenging gesture recognition tasks using raw MEM data. To validate the effectiveness of the network we conduct three sets of experiments. We achieved a classification accuracy with a the proposed network of about 90% which only 2% less than the full-precision network. We also compare our method with recent methods and outperform these methods by large margins on the conducted datasets.**

***Index Terms*—Indoor positioning, inertial tracking, dead reckoning, deep learning, machine learning**

I. Introduction

Deep Neural Networks (DNNs) have substantially pushed Artificial Intelligence in a wide range of tasks, including but not limited to object recognition from images [1], [2], speech recognition [3], [4] and even Atari and Go games [5], [6].

One field that has yet to benefit from deep learning is Human Activity Recognition (HAR) in Ubiquitous Computing (ubicomp). The dominant technical approach in HAR includes sliding window segmentation of time-series data captured with body-worn sensors, manually designed feature extraction procedures, and a wide variety of supervised classification methods [7]. In many cases, these relatively simple methods suffer to obtain the impressive recognition accuracies that neural networks achieved in the last decade.

Although DNNs are extending the state of the art results in the last decade: they are almost exclusively trained on one or many very fast and power-hungry Graphic Processing Units or on industrial-sized clusters [8], [9]. So, while they perform well on expensive, GPU-based machines or industrial-sized clusters, they are often unsuitable for smaller devices like cell phones and embedded electronics. For example, AlexNet [1] has 61M parameters (249MB of memory) and performs 1.5B high precision operations to make a prediction. These numbers are even higher for bigger networks with more parameters that call for more resources (processing power, memory, battery time, etc).

Another primary concern that hinders those applications from being more successful by using deep neural networks is that they assume an always-on pattern recognition engine on the device, which will drain the battery fast unless it is carefully implemented to minimize the use of resources.

The disjunction between these two trends creates a dilemma when state-of-the-art deep learning algorithms are designed to be deployed on mobile devices. This paper makes the following contributions to resolve the problems by introducing a simple, efficient, and accurate approximations to BLSTM-RNNs by binarizing the weights and inputs even in the intermediate layer of the neural network:

- We introduce a method to train and evaluate binarized BLSTM-RNNs (B-BLSTM-RNN) networks with binary weights and activations on embedded systems. Our approach draws on recent successes of BLSTM-RNNs, that demonstrated a minimal classification error on indoor localization related topics [10]. Such as step detection, step length estimation and classification of different movements to name a few.

- We conduct three sets of experiments, each performed on an embedded system, which show that it is possible to train and evaluate a binarized BLSTM-RNN (B-BLSTM-RNN) that achieve nearly state-of-the-art results.

- We show that during the forward pass (both at run-time and train-time), binarized BLSTM-RNNs drastically reduce memory consumption (size and number of accesses), and replace most arithmetic operations with bit-wise operations, which potentially lead to a substantial increase in power-efficiency.

The rest of the paper is organized as follows. Sec. II gives an overview of related works, followed by a detailed presentation of our new method to evaluate and train a recurrent neural network on embedded systems in sec. IV. In sec. IV-B we discuss the most important implementation issues and conclude with experiments in sec. V.

II. Related work

Efficient computational structures for deploying artificial neural networks have long been studied in the literature. Most of the effort is focused on training networks whose weights can be transformed into some quantized representations with a minimal loss of performance. The different types of networks are explained below.

Shallow Feed-forward networks: The main idea behind shallow networks is to train a compact model to approximate the function learned by a larger, more complex model. For example, in [11], [12], a single neural network is trained to mimic a much larger ensemble of models. The idea is based on the well-known early theoretical work on the representational capacity of neural nets. It was proven that a network with a large enough single hidden layer of sigmoid units can approximate any decision boundary [13]. Empirical work [10], [14], however, shows that it is difficult to train shallow nets to be as accurate as deep nets. In [10], [15], the authors show that deeper models are more competitive than shallow models in speech acoustic modeling. This method is different from our approach because we use the standard network architectures, not a shallow approximation.

Compressing pre-trained deep networks: Pruning redundant, non-informative weights in a previously trained network reduces the size of the network at inference time. Weight decay [16] was an early method for pruning a network. Optimal Brain Damage [17] use the Hessian of the loss function to prune a network by reducing the number of connections. Recently Wenlin Chen et al. introduced HashedNets [18] to reduce the number of parameters by using a low-cost hash function to randomly group connection weights into hash buckets, that share a single parameter value. Preetum Nakkiran [19] compressed an existing fully-trained DNN using a low-rank approximation of the weights associated with individual nodes in the first hidden layer by means of a rank-constrained DNN layer topology. They exploited the fact, that the weights, corresponding to the first hidden layer of the model, act as low-level feature detectors, and can thus be considered as filters. Filters tend to have a simple structure, which makes them amenable to compression. We are different from these approaches because we do not use a pre-trained network.

Network binarization: Several methods attempt to binarize the weights and the activations in neural networks. In [20], [21], the authors show that an approximation of all convolution operation in a convolutional network using primarily binary operations could result in a large speedup. In [22] Philipp Gysel et al. presents a complete model approximation framework that analyzes a given convolutional neural network. That transforms the floating point arithmetic into fixed point arithmetic with respect to numerical resolution used in representing weights and outputs of convolutional and fully connected layers. Our method is different from the mentioned method regarding the binarization method and the network structure, almost all of the proposed method are using a convolutional neural network as a basis. However, convolutional neural networks are difficult to be used in an activity classification task using MEM data, especially when using continues raw MEM data.

III. Deep Neural Networks

In this section, we present Binarized-BLSTM-RNN, a novel variation of the LSTM recurrent neural network with drastically reduced model sizes and memory demands. We first introduce the standard LSTM-RNN model and describe our approach to binarize the model using binary weights and activations to avoid any additional memory and performance overhead.

A. Deep Recurrent Networks

Deep recurrent networks, most notably those that rely on Long Short-Term Memory cells (LSTMs) [23], have recently achieved impressive performances across a variety of scenarios [24]. Their application to HAR has been explored in various settings [10]. Neverova et al. [25] investigated a variety of recurrent approaches to identify individuals based on movement data recorded from their mobile phone on a large dataset.

B. Long-Short Term Memory Networks

LSTM-RNNs are dynamical systems introduced by S. Hochreiter and J.Schmidhuber [23]. They have been successfully applied to a number of tasks in computer vision, bioinformatics, and natural language processing. In all of these applications, given an input sequence $x = (x_1, ..., x_T)$, a standard recurrent neural network computes the sequences of hidden vectors $h = (h_1, ..., h_T)$ and output vectors $z = (y_1, ..., y_T)$ by recursively evaluation the following equations from time steps $t = 1$ to $t = T$:

$$h^t = f_{act}(w_{xh}x_t + w_{hh}h_{t-1} + b_h) \quad (1)$$

$$z^t = w_{hy}h_t + b_y \quad (2)$$

w denote the weight matrices, b the bias vectors and f_{act} the activation function of the hidden layer, often chosen to be the sigmoid, tanh or sign function. Our binarized BLST-RNN uses the sign function. However, standard RNNs tend to suffer from the vanishing gradient problem [26], thus limiting their access to long time lags. Therefore, the Long Short-Term Memory (LSTM) model [23] was developed to better find and exploit long-range context using special memory cells.

In order to exploit the temporal dependencies within the movement data each LSTM cell (unit) keeps track of an internal state (the constant error carousel) that represents its ”memory”. Each memory block contains one or more recurrently connected memory cells and three multiplicative units, the input, output, and forget gates, which control the information flow inside the memory block. The surrounding network can only interact with the memory cells via the gates. In other words, these gates and the memory cell allow an LSTM unit to adaptively forget, memorize and expose the memory content. Leading to a network model capable of retaining information across hundreds of time-steps. One shortcoming of conventional RNNs is that they are only able to make use of the previous context. In motion detection especially in HAR, where an input with a fixed length is analyzed at once, there is no reason not to exploit future context as well. Bidirectional RNNs (BRNNs) [27] do this by processing the data in both directions. They use two separate hidden layers, which feed the data backwards to the same output layer. A BRNN computes the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence y by iterating the backward layer from $t = T$ to 1, the forward layer from $t = 1$ to T. At the end, the output layer is updated. Combining BRNNs with LSTM gives bidirectional LSTM [28] which can access long-range context in both input directions. By resorting to bidirectional networks true online processing is impossible, due to the need for a complete data sequence. However, for the action recognition task it is sufficient to obtain an output at the end of a fixed motion sequence. So that both passes, forward and backward, can be used during the recognition process. For the evaluation process described in this paper, we follow the implementation presented in [29].

IV. Binarized Neural Networks

Each iteration of training an RNN involves three steps; forward pass, backward pass, and parameter update. In this section, we detail our binarization function, show how we use it to compute the parameter gradients, and how we backpropagate through the network.

A. Training Bitwise Neural Networks

It has long been known that any boolean function, which takes binary values as input and produces binary outputs as well, can be represented as a bitwise network with one hidden layer [13], for example, by merely memorizing all the possible mappings between input and output patterns. Based on the idea, we train a BLSTM-RNN with binary weights. However, we only binarize the weights during the forward and backward pass. For updating the parameters, we use the high precision (real-value) weights. Because in gradient descend the parameter changes are tiny, binarization after updating the parameters ignores these changes and the training objective can not be improved. We also use the high precision (real-value) parameter for the LSTM-Layer in the forward pass as in the backward pass. Because the changes are also tiny and a representation using binary values, significant ignores these changes. Algorithm 1 demonstrates our procedure for training an RNN with binary weights.
First, we binarize the weights at each layer by computing $\mathcal{B}$ and α. Next we call the forward method using binary weights and its corresponding scaling factors for the non-LSTM layer and the high precision (real-value) weights for the LSTM layer. Afterwards, we call the backward algorithm, where the gradients are computed with respect to the estimated weights. Lastly, the network parameters and the learning rate are updated according to the architecture and optimizer. Note, that once the training finished, there is no need to keep the real-valued weights for the non-LSTM layer, since we only need all real-valued weights in the backward and update process.

Algorithm 1 Train an B-BLSTM-RNN with binary weights and binary activations

```
Input: input and target (I, T), weight $\mathcal{W}^t$
Output: updated weight $\mathcal{W}^{t+1}$:
1: Binarize weights $\mathcal{W}^t$:
2: $\alpha = \|\mathcal{W}^t\|$
3: $\mathcal{B} = sign(\mathcal{W}^t)$
4: $\widetilde{\mathcal{W}} = \alpha\mathcal{B}$
5: Forward($\mathcal{I}$, $\alpha$, $\mathcal{B}$, $\mathcal{W}^t$)  ▷ Standard forward propagation
   except that the weights and activations of the non-LSTM
   layer are binary.
6: Backward($\widetilde{\mathcal{W}}$, $\mathcal{E}$)  ▷ Standard backward propagation
   except that the gradient are computed with $\widetilde{\mathcal{W}}$ instead of
   $\mathcal{W}^t$.
7: UpdateParameters()  ▷ Update weights and learning
   parameter.
```

B. B-BLSTM-RNN

We follow the representation porposed by [30] and represent an L-layer BLSTM-RNN as a triplet $(\mathcal{I}, \mathcal{W}, *)$. Where L is the number of layer, $\mathcal{I}$ a set of vectors, where each element $\mathcal{I} = \mathcal{I}_{l(l=1,...,L)}$ is the input vector for the l'th layer of the B-BLSTM-RNN. $\mathcal{W}$ is a set of matrices, where each matrix $\mathcal{W} = \mathcal{W}_{l(l=1,...,L)}$ is the weight in the l'th layer of the network. Further, represents $*$ the dot-product between the binary input $\mathcal{I}$ and the binary weight $\mathcal{W}$. Note that in our description, we suppress the bias terms and sample time ($t = 1$ to $t = T$) for the notational convenience.

C. Estimating Binary Weights

In order to constrain a BLSTM recurrent neural network with binary weights, we estimate the real-value weights $\mathbf{W} \in \mathcal{W}$ using a binary weight $\mathbf{W} \in \{+1, -1\}$ and a scaling factor $\alpha \in \mathcal{R}^+$ such that $\mathbf{W} \approx \alpha\mathbf{B}$. So that the matrix multiplication between two layer can be approximated by:

$$\mathbf{I} * \mathbf{W} \approx (\mathbf{I} \oplus \mathbf{B})\alpha \tag{3}$$

We use bold characters for a vector or a matrix and non-bold characters for a scalar parameter, and $\oplus$ stands for the bitwise XNOR operation. For the approximation we follow the approach proposed by [30]:

$$\alpha = \|\mathcal{W}\| \tag{4}$$

$$\mathcal{B} = sign(\mathcal{W}) \tag{5}$$

In order to transform the real-valued variables into a binary representation, we use the deterministic sign functions as proposed in [31]:

$$z_i^l = sign(a_i^l) \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{6}$$

D. Estimating Binary Activations and Error

We define the binary activation procedure as follows based on the assumption that we have a network with binary weights:

$$a_i^l = b_i^l \sum_j^{K^{l-1}} w_{i,j}^l \oplus z_j^{l-1} \tag{7}$$

$$z_i^l = sign(a_i^l) \tag{8}$$

where $w_{i,j}^l \in \mathbf{W}$ and l indicates a layer and j, i the input and output units of an specific layer, respectively. K^l is the number of input units at the l'th layer. We can check the prediction error $\mathcal{E}$ by measuring the bitwise agreement of target vector t and the output units of l'th layer using XNOR as a multiplication operator:

$$\mathcal{E} = \sum_i^{K^{L+1}} \frac{(1 - t_i \oplus z_i^{L+1})}{2} \tag{9}$$

Note that we use $\{+1, -1\}$ as representation since it is more flexible when examining the network behavior, but when it comes to the implementation $+1$ stands for the "TRUE", while 1 stands for "FALSE" or $0/1$ respectively. So that the activation is equivalent to counting the number of 1's and then checking if the accumulation is bigger than half of the number of input units plus 1. With the result, that we can substitute any multiplication with faster binary multiplications.

V. Experiments

We now turn to present some experiments, which demonstrate the performance of the Binarized-BLSTM-RNN model. For comparison, we also implemented the standard BLSTM-RNN model, a Multilayer Perceptron (DNN), Dynamic Time Warping (DTW), and a Hidden-Markov model(MHMMR) , using several different training parameters, as well as a number of training iterations. Note that all algorithms were initialized from the same random vector, chosen uniformly at random to make the comparison as comparable as possible.

A. Experimental Setup

The next thing to consider is the empirical performance of the proposed model. In our experiments, we evaluated and compared the proposed model with the following methods:

B-LSTM-RNN & BLSTM-RNN: We use the same network architecture for the binarized and the standard LSTM recurrent neural network. The input data fed into the network corresponds to frames of movement data. Each frame consists of a number of s samples, which are simply concatenated into a single vector. Notice that both systems rely only on the calibrated sensor data, without any additional feature extraction in opposition to most state-of-the-art methods. These data are linearly normalized between -1 and $+1$ according to the maximum value that the sensors can provide. The forward and backward LSTM hidden layers are fully connected to the input layer and consist on five LSTM neurons each with full recurrent connections. The SoftMax activation function was used for the output-layer to give network responses between 0 and 1 at every time-step. Normally, these outputs can be considered as posterior probabilities of the input sequence to belong to a specific motion class at a given time-step. Following Zaremba et al. [32], we used the mixed curriculum strategy to model our network.

DNN: We implemented a deep feed-forward network, with five hidden layers followed by a softmax-layer. We have chosen the architecture so that the DNN based approaches have almost the same number of parameters as the RNN approach. Each hidden layer contains the same number of units and corresponds to a linear transformation and a rectified-linear activation function. We also used Dropout during the training phase. The input data fed into the network corresponds to frames of movement data. Each frame consists of a number of s samples, which are simply concatenated into a single vector. We minimized the negative log likelihood using RMSprop [33].

DTW: The DTW approach is an ensemble classifier based on Dynamic Time Warping (DTW) as proposed by D. McGlynn et al. [34]. The basic idea is to split up the training data for the system into a set of short time samples for each sensor and each activity, which are used as templates for the DTW approach. The time series for each sensor are classified by assessing their similarity to these templates. To get the final classification, results from separate classifiers are combined.

MHMMR: The Multiple Hidden Markov Model Regression (MHMMR) [35] approach uses the maximum likelihood as a decision rule as almost all HMM-based approaches. However, it's main advantage over traditional approaches, comes from the fact that the statistical model explains the regime changes over time through the hidden Markov chain, where each regime is interpreted as an activity.

VI. Datasets

We perform the experiment on several typical datasets and also created our own dataset for the performance comparison. Below we detail each dataset:

Opportunity dataset: The Opportunity dataset [36] contains human activities recorded from on-body sensors from 4 participants. Two types of recording sessions were performed. A Drill sessions where the subject performed a pre-defined set of activities 20 times (e.g. open/close the fridge or toggle the lights on/off) and a "daily living activities" run where the participants performed high-level task (wake up, groom, prepare breakfast, clean) with more freedom about the sequence of individual activities. We used a subset of the dataset and only used data from sensors that did not show any packet-loss. Which included accelerometer recordings from the upper limbs, the back, and complete IMU data from both feet. We used run 2 from subject 1 as our validation set, and replicate

dataset	**Opportunity**			**PAMAP2**			**Custom**		
configuration	C1	C2	C3	C1	C2	C3	C1	C2	C3
BLSTM-RNN	**0.74 ± 0.21%**	**0.78 ± 0.18%**	**0.78 ± 0.16%**	**0.90 ± 0.15%**	**0.93 ± 0.22%**	**0.93 ± 0.91%**	**0.79 ± 0.76%**	**0.81 ± 0.46%**	**0.83 ± 0.60%**
B-BLSTM-RNN	0.71 ± 0.33%	0.76 ± 0.16%	0.76 ± 0.15%	0.88 ± 0.21%	0.92 ± 0.21%	0.91 ± 0.94%	0.76 ± 0.54%	0.79 ± 0.33%	0.81 ± 0.44%
DNN	0.68 ± 0.40%	0.74 ± 0.90%	0.76 ± 0.27%	0.86 ± 0.21%	0.89 ± 0.23%	0.91 ± 0.81%	0.73 ± 0.51%	0.74 ± 0.33%	0.79 ± 0.81%
DTW	0.72 ± 0.71%	0.71 ± 1.15%	0.69 ± 0.15%	0.87 ± 2.01%	0.78 ± 0.98%	0.74 ± 0.99%	0.75 ± 1.20%	0.69 ± 0.93%	0.65 ± 0.94%
MHMMR	0.73 ± 0.62%	0.71 ± 2.11%	0.70 ± 0.91%	0.89 ± 1.01%	0.88 ± 0.99%	0.84 ± 0.99%	0.76 ± 2.20%	0.72 ± 0.92%	0.69 ± 1.94%

TABLE I: Recognition error and standard deviation, for individual configurations, datasets and methods.

the most popular recognition challenge by using runs 4 and 5 from subject 2 and 3 in our test set. The remaining data is used for training.

PAMAP2 dataset: The PAMAP2 Physical Activity Monitoring dataset [37] contains data of 18 different physical activities (e.g. walking, cycling, playing soccer), performed by 9 participants recorded by 3 inertial measurement units (Accelerometer, gyroscope, magnetometer) located on the hand, chest and ankle over and a temperature and heart rate sensor. We used run 1 and 2 for subject 5 in our validation set and run 1 and 2 for subject 6 in our test set. The remaining data is used for training.

Custom dataset: Our dataset [10] was obtained using a foot mounted IMU (3D-gyroscope, 3D-accelerometer, 3D-magnetometer). The ground truth data was obtained by manually labeling the data based on a down facing hand-held camera to note the exact time when the activity was performed and which activity was performed. 22 participants, from 20 to 55 years old, performed 15 times each, one of the 14 different activities (e.g. walking, running, open door). The sampling time for accelerometer and gyroscope capture is 100Hz. The 14 gestures are divided into 2 families: linear gestures (e.g. walking, running) and non-linear gestures (e.g. jumping, falling). These choices make the dataset difficult.

As the datasets are highly biased we require a performance metric that is independent of the class distribution. We estimate the mean f1-score:

$$F_m = \frac{2}{|c|} \sum_c \frac{precision_c \cdot recall_c}{precision_c + recall_c} \qquad (10)$$

where c is the current class and $|c|$ is the total number of classes.

A. Classification Results

We use 3 different configurations to compare our Binarized BLSTM-RNN to 4 to state-of-the-art solutions: BLSTM-RNN, DNN, DTW, and MHMMR. In all experiments, we use raw MEM information for the LSTM and DNN based solutions and gestural features for the DTW and MHMMR method. We also used a 3-fold cross-validation for all experiments. The first configuration (C1) corresponds to the personalization paradigm, where only one user is considered with few learning examples. For this configuration, we only used gestures of a single participant in the learning phase, and a subset of the same dataset for the test phase. The second configuration (C2) uses data from all participants in the learning phase and a subset of the same dataset for the test phase. This case corresponds to a multi-user system and a closed world paradigm. The third configuration (C3) is composed of all samples from all participants and the test data uses the other available gestures from unknown users, respectively data from participants that weren't used in the training phase. This case is close to a real system trained with a few examples; having to generalize to new users who want to use it without any personalization phase. This configuration represents the open world paradigm. Table I outlines the performance of each classifier for the different configurations.

The binarized BLSTM-RNN achieves almost every time the second best performance in every configuration. In fact, with the exception of minor residual noise introduced by the binarization process, the binarized BLSTM-RNN is almost identical to the non-binarized BLSTM-RNN model. Another observation is that the binarized BLSTM-RNN algorithm tends to perform better than any non-recurrent algorithm evaluated, despite the fact that we used binary parameter and high precision (real-value) parameter for the internal LSTM operations. There are multiple potential explanations for this phenomenon:

- The network architecture mainly relies on the high precision LSTM-cell, to achieve high accuracies.
- The model is able to generalize even with binarized weights and parameters.

Note that the binarized BLSTM-RNN achieves the second best performance in almost every configuration and outperforms the other method by a larger margin.

B. Computing Times

The results for the computing times (mean over 10 trails) are summarized in table II and detailed for the custom dataset. The table also provides the standard deviation of the computing times during the evaluation for the proposed model B-BLSTM-RNN, BLSTM-RNN, DNN, DTW, and MHMMR. As would be expected for the different algorithms. The prediction method for all deep learning and HMM-based approaches aren't bound to the dataset size, so larger datasets (such as C2) tend to provide larger speedups compared to the smaller datasets (C1). On the contrary, DTW needs to compare the input gesture with all learned references samples. That is why the computing time increases in mean from 14.45 ms for C1 (less samples) to 51.71ms for C3 (more learning sample). Consequently, the Binarized BLSTM-RNN achieves the best performances in the multi-user configuration with a computing time independent from the training dataset size.

configuration	**C1**	**C2**	**C3**
B-BLSTM-RNN	**13.34 $\pm$ 0.04**	**12.89 $\pm$ 0.09**	**13.98 $\pm$ 0.09**
BLSTM-RNN	54.32 $\pm$ 0.02	52.07 $\pm$ 0.02	51.22 $\pm$ 0.06
DNN	35.12 $\pm$ 0.03	34.89 $\pm$ 0.06	35.09 $\pm$ 0.03
DTW	14.45 $\pm$ 0.02	39.53 $\pm$ 0.45	51.71 $\pm$ 0.35
MHMMR	44.53 $\pm$ 1.84	25.91 $\pm$ 2.42	33.19 $\pm$ 1.44

TABLE II: Computing times / prediction times (in ms) for the custom dataset, individual configurations and methods to classify one unknown activity.

Another observation is that the proposed method scales better (in terms of runtime and memory) than the other algorithms. These results show that our algorithm satisfies its original goals: to be able to scale effectively in resource-constrained situations, particularly in cases where floating-point / fixed-point variables and operations are prohibitively expensive.

VII. Conclusion and Outlook

One of the major shortcomings when using state-of-the-art deep neural networks for activity recognition is the lack of resources in embedded and mobile systems. By explicitly addressing this shortcoming we have demonstrated the suitability of using a binary representation of the BLSTM recurrent neural network. Leading to a system, in which both the activations and the weights are constrained to either 1 or $+1$. With the result, that most of the floating point multiply-accumulations are replaced by 1-bit XNOR-count operations. Since our proposed binarized recurrent model is flexible, powerful, and efficient, we believe that it may be a promising approach for tackling other challenging tasks on power and performance constrained systems.

Despite this, there are still several extensions and improvements that may be performed but are not described in detail here:

- Provable worst-case runtime bounds. A relationship between the properties of the model architecture and the runtime may be derived, from the binarization strategy and data size.
- Focus on the imbalance problem caused by the distortions existing in the dataset as well as improving the sensor data calibration.
- Investigate the effect of feature selection more deeply, which might lead to a more generalized system.
- Perform more in-depth experiments with Binarized BLSTM-RNNs and combinations of Binarized BLSTMs with convolutional neural networks. Following the goal to improve the overall system accuracy by combining all advantages.

Acknowledgment

The research presented in this publication was conducted while the first author was employed by the Federal Institute for Materials Research and Testing.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, 2014.

[3] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for lvcsr," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 2013, pp. 315–320.

[4] D. Yu and L. Deng, *Automatic Speech Recognition - A Deep Learning Approach*. Springer, Oct. 2014.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, 2013.

[6] "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[7] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, no. 3, 33:1–33:33, Jan. 2014.

[8] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[9] A. Coates, B. Huval, T. Wang, D. J. Wu, B. C. Catanzaro, and A. Y. Ng, "Deep learning with COTS HPC systems," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 2013, pp. 1337–1345.

[10] M. Edel and E. Köppe, "An advanced method for pedestrian dead reckoning using blstm-rnns," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, 2015, pp. 1–6.

[11] K. Wang, X. Wang, L. Lin, M. Wang, and W. Zuo, "3d human activity recognition with reconfigurable convolutional neural networks," *CoRR*, 2015.

[12] S. Oniga and J. Sütö, "Human activity recognition using neural networks," in *Control Conference (ICCC), 2014 15th International Carpathian*, May 2014, pp. 403–406.

[13] W. S. McCulloch and W. Pitts, "Neurocomputing: Foundations of research," in, J. A. Anderson and E. Rosenfeld, Eds., MIT Press, 1988, ch. A Logical Calculus of the Ideas Immanent in Nervous Activity, pp. 15–27.

[14] Y. Chen and Y. Xue, "A deep learning approach to human activity recognition based on single accelerometer," in *SMC*, IEEE, 2015, pp. 1488–1492.

[15] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *Ijcai*, 2016.

[16] S. J. Hanson and L. Pratt, "Advances in neural information processing systems 1," in, D. S. Touretzky, Ed., Morgan Kaufmann Publishers Inc., 1989, ch. Comparing Biases for Minimal Network Construction with Back-propagation, pp. 177–185.

[17] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, Morgan Kaufmann, 1990, pp. 598–605.

[18] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," *CoRR*, 2015.

[19] P. Nakkiran, R. Alvarez, R. Prabhavalkar, and C. Parada, "Compressing deep neural networks using a rank-constrained topology," in *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, 2015, pp. 1473–1477.

[20] M. Kim and S. Paris, "Bitwise neural networks," *ICML Workshop on Resource-Efficient Machine Learning*, vol. 37, 2015.

[21] M. Courbariaux and Y. Bengio, "Binarynet: training deep neural networks with weights and activations constrained to +1 or -1," 2016.

[22] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," p. 8, 2016.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[24] R. Józefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 2342–2350.

[25] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *Ijcai*, 2016.

[26] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in, Kremer and Kolen, Eds., IEEE Press, 2001.

[27] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[28] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[29] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *CoRR*, 2013.

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *CoRR*, 2016.

[31] M. Courbariaux, Y. Bengio, and J. David, "Low precision arithmetic for deep learning," *CoRR*, 2014.

[32] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, 2014.

[33] T. Tieleman and G. Hinton, *Lecture 6.5—rmsprop: divide the gradient by a running average of its recent magnitude*, COURSERA: Neural Networks for Machine Learning, 2012.

[34] D. McGlynn and M. G. Madden, "Research and development in intelligent systems xxvii: Incorporating applications and innovations in intelligent systems xviii proceedings of ai-2010, the thirtieth sgai international conference on innovative techniques and applications of artificial intelligence," in, M. Bramer, M. Petridis, and A. Hopgood, Eds. London: Springer London, 2011, ch. An Ensemble Dynamic Time Warping Classifier with Application to Activity Recognition, pp. 339–352.

[35] D. Trabelsi, S. Mohammed, F. Chamroukhi, L. Oukhellou, and Y. Amirat, "An unsupervised approach for automatic activity recognition based on hidden markov model regression.," *CoRR*, 2013.

[36] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. del R. Millán, "Collecting complex activity datasets in highly rich networked sensor environments," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, Jun. 2010, pp. 233–240.

[37] A. Reiss and D. Stricker, "Creating and benchmarking a new dataset for physical activity monitoring," in *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '12, New York, NY, USA: ACM, 2012, 40:1–40:8.