An Efficient Video Prediction Recurrent Network using Focal Loss and Decomposed Tensor Train for Imbalance Dataset

Mingshuo Liu
California State University, Fullerton
liumingshuo@csu.fullerton.edu

Kevin Han
California State University, Fullerton
khchocosj@csu.fullerton.edu

Shiyi Luo
California State University, Fullerton
luoshiyi@csu.fullerton.edu

Mingze Pan
California State University, Fullerton
pm2.5@csu.fullerton.edu

Mousam Hossain
University of Central Florida
mousam.hossain@knights.ucf.edu

Bo Yuan
Rutgers University
bo.yuan@soe.rutgers.edu

Ronald F. DeMara
University of Central Florida
Ronald.Demara@ucf.edu

Yu Bai
California State University, Fullerton
ybai@fullerton.edu

ABSTRACT

Nowadays, from companies to academics, researchers across the world are interested in developing recurrent neural networks due to their incredible feats in various applications, such as speech recognition, video detection, prediction, and machine translation. However, the advantages of recurrent neural networks accompanied by high computational and power demands, which are a major design constraint for electronic devices with limited resources used in such network implementations. Optimizing the recurrent neural networks, such as model compression, is crucial to ensure the broad deployment of recurrent neural networks and promote recurrent neural networks for implementing most resource-constrained scenarios. Among many techniques, tensor train (TT) decomposition is considered an up-and-coming technology. Although our previous efforts have achieved 1) expanding the limits of many multiplications eliminating all redundant computations; and 2) decomposing into multistage processing to reduce memory traffic, this work still faces some limitations. In particular, current TT decomposition on recurrent neural networks leads to a complex computation sensitive to the quality of training datasets. In this paper, we investigate a new method for TT decomposition on recurrent neural networks for constructing an efficient model within imbalance datasets to overcome this issue. Experimental results show that the proposed new training method can achieve significant improvements in accuracy, precision, recall, F1-score, False Negative Rate (FNR), and False Omission Rate (FOR).

GLSVLSI '21, June 22–25, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8393-6/21/06…$15.00
https://doi.org/10.1145/3453688.3461748

CCS CONCEPTS

• **Computing methodologies** → **Activity recognition and understanding**; • **Computer systems organization** → **Embedded hardware**.

KEYWORDS

Tensor decomposition, focal Loss, embedded hardware

ACM Reference Format:
Mingshuo Liu, Kevin Han, Shiyi Luo, Mingze Pan, Mousam Hossain, Bo Yuan, Ronald F. DeMara, and Yu Bai. 2021. An Efficient Video Prediction Recurrent Network using Focal Loss and Decomposed Tensor Train for Imbalance Dataset. In *Proceedings of the Great Lakes Symposium on VLSI 2021 (GLSVLSI '21), June 22–25, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3453688.3461748

1 INTRODUCTION

Deep learning research in advancing sequence modeling has been drawing more attention from both academia and industry [2, 9, 21]. Among various research efforts, Long Short-Term Memory (LSTM) is considered as a promising recurrent neural network that employs gates to control information flow in recurrent computations [4, 7, 23]. The LSTM model provides a powerful tool in various tasks such as machine translation, language modeling, health information, time series prediction, and speech [1, 8, 12, 14, 22, 24]. However, current LSTM can not efficiently be employed in resource-constrained applications due to its large model size.

To address this issue, *model pruning and compression* techniques have been developed to construct compact LSTM models, and its efficient hardware accelerators [3, 5, 10, 17, 19, 26–29]. Among these compression technologies, *tensor train (TT) decomposition* is an attractive mathematical tool to decompose a large-scale tensor to a group of smaller tensor cores with an efficient compression of any tensor formats. However, current TT decomposition methods still face two challenging limitations: 1) TT-decomposition increases the number of possible layer-wise quantization schemes since it decomposes one large tensor to multiple small tensor cores; 2) some limit memory devices such as FPGA can not adopt and perform efficient LSTMs on-chip. The current TT-decomposition on LSTM also faces an additional challenge to train an efficient LSTM model

from an imbalanced dataset with an insufficient number of images on certain categories in real-world applications. Consequently, the performance of TT-decomposition enhanced LSTM is significantly reduced.

A novel approach to construct an optimization LSTM that performs accurate prediction for an imbalance dataset within a high-performance LSTM hardware accelerator is developed in this paper to overcome these issues. Specifically, the main technical contributions are as follows:

- In this paper, we have developed a novel approach named Focal Loss-Tensor-Train LSTM that can train an efficient LSTM model within an imbalance dataset. Compared with prior TT-based LSTMs, our proposed TT-LSTMs achieve an improvement in model performance.
- At the hardware design level, we have developed a novel architecture of the hardware accelerator based on our proposed FL-TT-LSTM to maximize the parallelism and the processing throughput of our FL-TT-LSTM accelerator.

The remainder of this paper is organized as follows: Section 2 introduces the related background. Section 3 proposes the new training method for improving its performance on imbalanced datasets. Section 4 shows experimental results using the proposed method to analyze software and hardware performance. Finally, Section 5 concludes the manuscript.

2 RELATED WORK

2.1 Long Short Term Memory (LSTM) networks

Long short-term memory(LSTM) is an artificial circulation neural network designed for time series prediction in deep learning without losing effectiveness with long delay [20]. LSTM can process and predict important events with very long intervals and delays in time series. Recent research efforts apply LSTM in human action recognition from video sequences. An LSTM unit consists of an input gate, an output gate, a forget gate, and a cell, as shown in Fig. 1. The cell stores the value for an indefinite length of time, and the three gates regulate the flow of information into and out of the cell. The forget gate in the LSTM unit is controlled by a neuron. It can decide the retaining or discarding of information to store historical information. The input gate is generated by the current neuron and the previous memory unit, which can determine whether to update the historical information to the LSTM block. The activation of each gate and the current cell status updated over time are calculated as follows:

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned}
\tag{1}
$$

where σ, tanh and $\odot$ are the sigmoid function, hyperbolic function, and element-wise product, respectively. The $\boldsymbol{f}$, $\boldsymbol{i}$, $\boldsymbol{c}$, $\boldsymbol{o}$ and $\boldsymbol{h}$ denote the forget gate, input gate, cell status, output gates, and hidden node. Additionally, $\mathbf{W}$ and $\mathbf{b}$ represent the weight and bias.

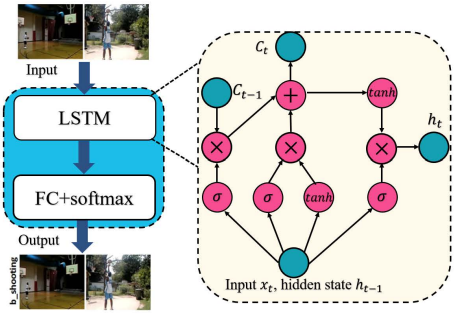

Figure 1: Architecture of the LSTM model used in this paper and LSTM unit structure

2.2 Tensor Decomposition

Novikov et al. [18] proposes the first Tensorizing Neural Networks. The weight of layers could determine the number of neural network parameters and Tensor Neural Network stores the weights in tensor format. Therefore, compressing the model by tensor decomposition will save the storage space and boost the calculation speed. Garipov et al. [6] rehearse the tensor train based training method with convolutional and FC layers. The size of their model has been cut down more than a dozen or even dozens of times based on different TT - ranks for less than 0.1 percent accuracy rate dropped. Tjandra et al. [25] have introduced tensor-train on RNN. Subsequently, Ye et al. [28] make the model more competitive by using the block-term decomposition method. He et al. [11] extend the tensor train method to the LSTM network that can share parameters and time calculation to perform the depth calculation of sequential tasks. Remarkable results are also presented in the language modeling task [16].

Herein, we define boldface letters $\boldsymbol{a}$, $\boldsymbol{A}$ and $\boldsymbol{\mathcal{A}}$ as vector, matrix, and tensor, respectively. *Tensor-Train Decomposition*: Given a d-dimensional tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ with a size of $n_1 \times n_2 \times \ldots \times n_d$, it can be decomposed as:

$$
\boldsymbol{\mathcal{A}}(i_1, \cdots, i_d) = \sum_{k_0, k_1, \cdots, k_d}^{r_0, r_1, \cdots, r_d} \mathcal{G}_1(k_0, i_1, k_1) \cdots \mathcal{G}_d(k_{d-1}, i_d, k_d), \tag{2}
$$

where the group of $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is a group of TT-cores and the sequence $\{r_k\}_{k=0}^{d}$ is defined as the TT-ranks. Parameters r_0 and r_d are set to 1. Once a large tensor $\boldsymbol{\mathcal{A}}$ is decomposed to the TT-format, the storage complexity is bounded by dnr^2, where $r = \max\{r_k\}_{k=0}^{d}$, $n = \max\{n_k\}_{k=1}^{d}$.

3 PROPOSED FOCAL LOSS ENHANCED TENSOR-TRAIN BASED INFERENCE

3.1 Tensor-Trained LSTM (TT-LSTM)

In LSTM structure, the linear layer is a basic element that outputs $\boldsymbol{y} \in \mathbb{R}^M$ which can be obtained by:

$$
\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}, \tag{3}
$$

where $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ is the weight matrix, $\boldsymbol{x} \in \mathbb{R}^N$ is the input vector. **Tensorization** Herein, we present a tensorization for the 2-D weight matrix to perform TT-decomposition in the linear layer. Given a weight matrix $\boldsymbol{W}$, it can be tensorized to a weight tensor

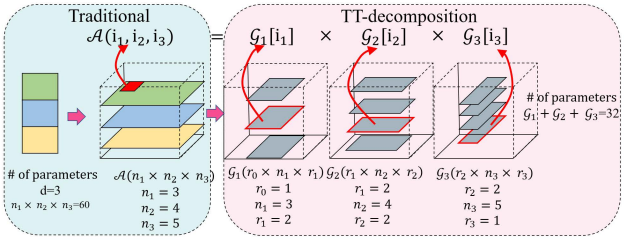

Figure 2: TT decomposition method for representing the matrix with reshaped tensor.

$\mathcal{W} \in \mathbb{R}^{(m_1 \times n_1) \times \cdots \times (m_d \times n_d)}$ by reshaping dimensions and transposing orders. Similar to the decomposition of weight matrix, input vector $\boldsymbol{x}$ and the output vector $\boldsymbol{y}$ also can be reshaped by factorization $M = \prod_{k=1}^{d} m_k, N = \prod_{k=1}^{d} n_k$ to a new form of the input tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and the output tensor $\mathcal{Y} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$, respectively.

Weights Decomposition In previous steps, all variables are performed tensorization. Consequently, all weights $\mathcal{W}$ are stored into small-size tensors that can be defined by TT-format (2) in element-wise form. $\mathcal{W}$ is decomposed as

$$\mathcal{W}((i_1, j_1), \cdots, (i_d, j_d)) = \sum_{k_0, k_1, \cdots, k_d}^{r_0, r_1, \cdots, r_d} \mathcal{G}_1(k_0, i_1, j_1, k_1) \cdots \mathcal{G}_d(k_{d-1}, i_d, j_d, k_d). \tag{4}$$

With one dimension more than the standard TT-cores in (2), here each $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times m_k \times r_k}$ is a 4-dimensional tensor, since there are factorized dimensions n_k and m_k in the tensorized $\mathcal{W}$.

TT-LSTM The TT-based LSTM is built by decomposing the input-to-hidden layer into smaller TT-format using *Tensor Train Layer* (*TTL*) method, as in [27]. Specifically, with input-to-hidden weight matrix $\boldsymbol{W} \in \mathbb{R}^{M \times N}$, hidden-to-hidden weight matrix $\boldsymbol{U} \in \mathbb{R}^{M \times M}$ and bias vector $\boldsymbol{b} \in \mathbb{R}^{M}$, the TT-LSTM can be represented as

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(TTL(\boldsymbol{W}_i, \boldsymbol{x}_t) + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(TTL(\boldsymbol{W}_f, \boldsymbol{x}_t) + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \\
&\quad \boldsymbol{i}_t \odot \tanh(TTL(\boldsymbol{W}_c, \boldsymbol{x}_t) + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \\
\boldsymbol{o}_t &= \sigma(TTL(\boldsymbol{W}_o, \boldsymbol{x}_t) + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{5}$$

where σ, tanh and $\odot$ are the sigmoid function, hyperbolic function, and element-wise product, respectively. The formulated TT-LSTM has the compression ratio (CR) given by

$$CR = \frac{MN + M^2}{\sum_{k=1}^{d} m_k n_k r_{k-1} r_k + M^2}, \tag{6}$$

where $M = \prod_{k=1}^{d} m_k, N = \prod_{k=1}^{d} n_k$. In Fig. 2, we present an example to store a 5×12 weight matrix in the TT format. The given $\boldsymbol{A}$ matrix and its reshaped 3-dimensional tensor $\mathcal{A}(i_1, i_2, i_3)$, can be decomposed and stored as three tensor cores $\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \boldsymbol{\mathcal{G}}_3$. Compared with traditional methods, our TT-decomposition can significantly reduce the number of parameters in computation.

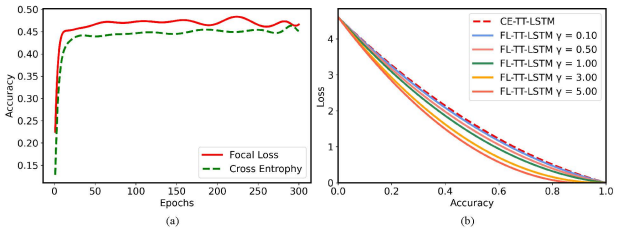

Figure 3: (a) The training accuracy comparison between FL and CE within different epochs, and (b) Demonstration of reduction in the model loss due to the addition of new factor $(1 - p_t)^{\gamma}$ and increasing of γ. As shown in this experiment, the FL model can provide more accurate training compared to the CE model.

3.2 Focal Loss Enhanced Tensor-Trained LSTM (FL-TT-LSTM)

A common method of loss function Cross Entropy (CE), which is widely used in most ML systems [13]. The traditional Cross Entropy (CE) loss given by (7), assigns a small weight to samples that are easy to be classified, and greater weight to more complex samples that are difficult to distinguish. Cross entropy function:

$$CE(p_t) = -log(p_t) \tag{7}$$

where

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & otherwise \end{cases}$$

and p is the model estimation probability for the class with the label $y = 1$. When the confidence is greater than 0.5, the value of the loss is not small. If there are many negative samples, the accumulation of these large losses will have an impact on class training with small samples. A simple approach is to assign different weights α to different classes, i.e., α-balanced cross entropy. In practice, α belongs to a default hyper-parameter, and the larger the sample number of the category, the smaller the setting of α.

$$CE(p_t) = -\alpha_t log(p_t) \tag{8}$$

The α-balanced cross entropy only balances the weights according to the number of positive and negative samples, without considering the difficulty of the samples. Therefore, the weight of easily classified samples is reduced so that the model is more focused on the samples that are difficult to classify during training.

Thus, the cross entropy needs to be redefined by adding a modulating factor $(1 - p_t)^{\gamma}$ with a tunable focusing parameter $\gamma \geq 0$. Consequently, the new loss function named Focal Loss as:

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \tag{9}$$

In some practices, we use an α-balanced variant of the focal loss:

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} log(p_t) \tag{10}$$

By conducting experiments on different tasks, the focal loss with α-balanced achieves better performance. To test the FL method to obtain its performance compared to the CE loss function, we present a loss comparison between FL and CE within the same dataset, see

Fig. 3. This experiment shows that the proposed FL applied to LSTM provides a high accuracy training that outperforms the traditional training method. We implement the proposed FL-TT-LSTM in the TensorFlow-Keras framework. Tensorflow framework is known as a flexible and powerful framework to support machine learning development, and Keras is a high-level API that can call Tensorflow as a backend. The details of the method to construct a new training method for FL-TT-LSTM can be found in Algorithm 1.

Algorithm 1 Focal-Loss implementation in TT-LSTM

Input: Weight $\mathcal{W}$, Train Dataset T, # Batches N
Hyparameters: Batch size B, Learning Rate η, training iterations *Epochs*
Output: $\mathcal{W}_i$

```
W = initialization(W)
CE = −log(p_t)
for i in range (Epochs) do
    for j in range (N) do
        MiniBatch = T[j × B : (j + 1) × B]
        Gs, bias, recurrent_kernel = W[0], W[1], W[2]
        res = MiniBatch
        for G in Gs do
            res = Reshape(res) · Reshape(G)
        end for
        preds = Dense(LSTM(res + h · recurrent_kernel + bias))
        p_t = Pred_Judgment(labels, preds)
        FL = α(1 − p_t)^γ CE(p_t)
        ∇FL(W_{i−1}) = Backward(FL, MiniBatch)
        W_i = W_{i−1} - η(1/B ∇FL(W_{i−1}))
    end for
end for
```

3.3 Overall Architecture of FPGA Accelerator for FL-TT-LSTM

In Fig. 4, we present our proposed overall architecture of FPGA Accelerator for FL-TT-LSTM. The architecture consists of computation kernel, registers, and BRAM. Due to the algorithm, unlike the traditional gigantic weight matrices requiring massive exterior storage, the FL-TT-LSTM Core weight is able to fit in the onboard BRAM and access via DMA and bus line results in a massive transmitting speed improvement. A PE (processing element) array is combined with the DSP as the multiply-accumulate (MAC), fulfilling a large matrix multiplication in the LSTM algorithm. Attributed to the FL-TT-LSTM, the parameters of the LSTM are reduced significantly, leading to greater resource efficiency of the logic component. In Fig. 4, the inference routing component is in charge of all the algorithm scheduling, the basic matrix multiplication, address update, and activation process. With a relatively smaller scale of computation in each row of the weight matrices, the flexibility of pipelining is foreseeable with better FPGA chips. On the other hand, with the strength of the FT-TT-LSTM, implementing on a low power edge computing device is also a rewarding possibility.

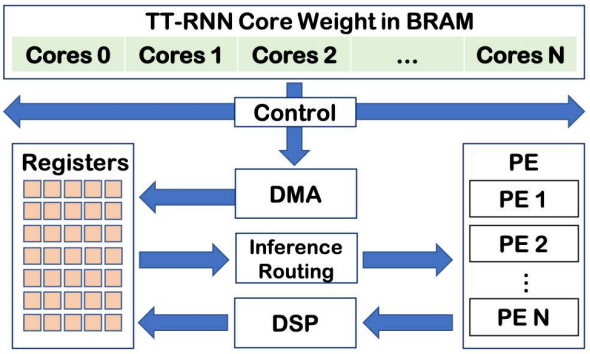

Figure 4: Architecture of FPGA Accelerator for FL-TT-LSTM

4 EXPERIMENT AND RESULTS

In this study, we propose the FL-TT-LSTM network to conduct the classification of the imbalanced UCF11 dataset compared with TT-LSTM model with cross entropy (CE) loss function. We show that the network of FL-TT-LSTM still has superior performance even in extreme imbalance. It is proved that Focal loss(FL) can improve the classification effect of the imbalanced dataset by comparing it with state-of-the-art methods.

4.1 Experimental Setup

We built a TT-LSTM with Cross Entropy(CE) loss as the benchmark, and the architecture is the same as the TT-LSTM [27] in Fig. 1. The second model where the Focal Loss (FL) is used to substitute the CE loss termed as FL-TT-LSTM. Both the TT-LSTM and the FL-TT-LSTM model are deployed in Keras, running in the same environment to ensure the validity of the comparison. In the following experiment, we set the epoch to 300. Both the TT-LSTM and the FL-TT-LSTM network have the same input: the concatenation of flattened frames that we sampled from every video clip. The input dimension is $159 \times 128 \times 3 = 61056$, factorized as $16 \times 8 \times 9 \times 53$, the hidden layer as $4 \times 4 \times 4 \times 6 = 384$ and the ranks are $[1, 4, 4, 4, 1]$. The computer configuration is AMD Ryzen 7 3700x + GTX1660s. By observing the curve in Fig. 3a), the overall classification accuracy of FL-TT-LSTM shows a good performance across the training process.

4.2 Comparison to State-of-the-Art in UCF11 Dataset

UCF YouTube Action Dataset (UCF11) [15], as a human action video dataset, contains 1600 video clips, and the resolution is 320 × 240. It is divided into 11 Action categories, such as swinging, riding a horse, playing basketball, and so on and its content is more consistent with the display, such as the lighting level, camera shake, background clutter, etc. We sample 6 frames from every video clips and generate a series of RGB frames with the size of 159×128 from each clip.

In this experiment, we analyzed the impact of imbalance ratio and critical parameters γ and α for FL on the classification task of the proposed TT-LSTM with CE and TT-LSTM network with FL. The Imbalance Ratio (IR) is defined as:

Table 3: Implementation speedup comparison with the baseline LSTM

	LSTM	Ours
Platform	XCKU5P	XCKU5P
Frequency(MHz)	200	200
Parameters	94.38M	0.60M
Matrix Compression Rate	1	28K×
Accuracy	79.9%	90.6%
Throughput (GOPS)	86.6	175.6
Power(W)	29.3	18.6
Power Efficiency (GOPS/W)	2.96	9.44

5 CONCLUSION

This paper develops a new method to construct an efficient LSTM inference accelerator utilizing an FPGA board. Through our proposed the new method, the proposed FL-TT-LSTM accelerator, can achieve 90.6% in accuracy compared to state-of-the-art 79.6%. On the heaviest imbalanced dataset, the proposed FL-TT-LSTM can achieve 49.0% in accuracy, 57.8% in precision, 54.7% in recall, 46.2% in F1-score, 0.0492 in FOR, and 0.4529 in FNR. In the hardware aspect, our proposed FL-TT-LSTM accelerator can achieve 18.6W power consumption and 9.44 power efficiency (GOPS/W) on different workloads in general. Compared to baseline-LSTM, it achieves 2.03× higher throughput, 10.7% accuracy increase, 3.2× power efficiency, and 1.58× power reduction. Our experimental results show that the proposed method enables the LSTM can achieve a remarkable improvement in the imbalance dataset with significant advantages in hardware over state-of-the-art solutions.

ACKNOWLEDGMENT

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-20-1-0174. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. This project was also partially supported by National Science Foundation under Grant CCF-1955909.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Y. Bai, D. Fan, and M. Lin. 2018. Stochastic-Based Synapse and Soft-Limiting Neuron with Spintronic Devices for Low Power and Robust Artificial Neural Networks. *IEEE Transactions on Multi-Scale Computing Systems* 4, 3 (2018), 463–476. https://doi.org/10.1109/TMSCS.2017.2787109

[3] Yuxuan Cai, Hongjia Li, Geng Yuan, Wei Niu, Yanyu Li, Xulong Tang, Bin Ren, and Yanzhi Wang. 2020. YOLObile: Real-Time Object Detection on Mobile Devices via Compression-Compilation Co-Design. *arXiv preprint arXiv:2009.05697* (2020).

[4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[5] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, Xiaolong Ma, Yipeng Zhang, Jian Tang, Qinru Qiu, Xue Lin, and Bo Yuan. 2017. CirCNN: Accelerating and Compressing Deep Neural Networks Using Block-CirculantWeight Matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, Massachusetts) *(MICRO-50 '17)*. Association for Computing Machinery, New York, NY, USA, 395–408. https://doi.org/10.1145/3123939.3124552

[6] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. 2016. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214* (2016).

[7] Tian Guo, Tao Lin, and Nino Antulov-Fantulin. 2019. Exploring interpretable LSTM neural networks over multi-variable data. In *International Conference on Machine Learning*. PMLR, 2494–2504.

[8] Tian Guo, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya. 2016. Robust online time series prediction with recurrent neural networks. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Ieee, 816–825.

[9] Yanhui Guo, Siming Han, Chuanhe Shen, Ying Li, Xijie Yin, and Yu Bai. 2018. An Adaptive SVR for High-Frequency Stock Price Forecasting. *IEEE Access* 6 (2018), 11397–11404. https://doi.org/10.1109/ACCESS.2018.2806180

[10] Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic network surgery for efficient dnns. *arXiv preprint arXiv:1608.04493* (2016).

[11] Zhen He, Shaobing Gao, Liang Xiao, Daxue Liu, Hangen He, and David Barber. 2017. Wider and deeper, cheaper and faster: Tensorized lstms for sequence learning. *arXiv preprint arXiv:1711.01577* (2017).

[12] Nan Rosemary Ke, Konrad Żołna, Alessandro Sordoni, Zhouhan Lin, Adam Trischler, Yoshua Bengio, Joelle Pineau, Laurent Charlin, and Christopher Pal. 2018. Focused hierarchical rnns for conditional sequence processing. In *International Conference on Machine Learning*. PMLR, 2554–2563.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[14] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. 2015. Learning to diagnose with LSTM recurrent neural networks. *arXiv preprint arXiv:1511.03677* (2015).

[15] Jingen Liu, Jiebo Luo, and Mubarak Shah. 2009. Recognizing realistic actions from videos "in the wild". In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1996–2003.

[16] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Dawei Song, and Ming Zhou. 2019. A tensorized transformer for language modeling. *arXiv preprint arXiv:1906.09777* (2019).

[17] Chuhan Min, Aosen Wang, Yiran Chen, Wenyao Xu, and Xin Chen. 2018. 2pfpce: Two-phase filter pruning based on conditional entropy. *arXiv preprint arXiv:1809.02220* (2018).

[18] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. 2015. Tensorizing neural networks. *arXiv preprint arXiv:1509.06569* (2015).

[19] Yu Pan et al. 2019. Compressing recurrent neural networks with tensor ring for action recognition. In *AAAI*, Vol. 33. 4683–4690.

[20] Hasim Sak, Andrew W Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. (2014).

[21] A. Samiee, P. Borulkar, R. F. DeMara, P. Zhao, and Y. Bai. 2019. Low-Energy Acceleration of Binarized Convolutional Neural Networks using a Spin Hall Effect based Logic-in-Memory Architecture. *IEEE Transactions on Emerging Topics in Computing* (2019), 1–1. https://doi.org/10.1109/TETC.2019.2915589

[22] Ashkan Samiee, Yinjie Huang, and Yu Bai. 2018. FRLDM: Empowering K-nearest Neighbor (KNN) through FPGA-based Reduced-rank Local Distance Metric. In *2018 IEEE International Conference on Big Data (Big Data)*. 4742–4746. https://doi.org/10.1109/BigData.2018.8622087

[23] Jürgen Schmidhuber and Sepp Hochreiter. 1997. Long short-term memory. *Neural Comput* 9, 8 (1997), 1735–1780.

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215* (2014).

[25] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2017. Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 4451–4458.

[26] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665* (2016).

[27] Yinchong Yang et al. 2017. Tensor-train recurrent neural networks for video classification. In *ICML*. 3891–3900.

[28] Jinmian Ye et al. 2018. Learning compact recurrent neural networks with block-term tensor decomposition. In *CVPR*. 9378–9387.

[29] Miao Yin, Siyu Liao, Xiao-Yang Liu, Xiaodong Wang, and Bo Yuan. 2020. Compressing recurrent neural networks using hierarchical tucker tensor decomposition. *arXiv preprint arXiv:2005.04366* (2020).