Real-time Anomaly Detection for Flight Testing using AutoEncoder and LSTM

Zhiqiang Que*, Yanyang Liu†, Ce Guo*, Xinyu Niu§, Yongxin Zhu‡, Wayne Luk*
*Dept. of Computing, Imperial College London, UK, {z.que, c.guo, w.luk}@imperial.ac.uk
† Lingnan Big Data Institute, China, {liuyanyang}@chiefdata.org
§ Corerain Technologies Ltd., China, {xinyu.niu}@corerain.com
‡ Shanghai Advanced Research Institute, Chinese Academy of Sciences, China, {zhuyongxin}@sari.ac.cn

***Abstract*—Flight testing is crucial in validating the functionality and safety in new commercial aircraft design before mass production. The challenge is to support real-time analysis of high-dimensional time series data generated from tens of thousands of sensors around the aircraft during test flights. We propose a novel 2-stage approach, using a fine-tuned autoencoder to extract the generic underlying features of high-dimensional data, followed by a stacked LSTM using the learned features to predict aircraft time series and to detect anomalies in real-time for flight testing. A novel Timestep(TS)-buffer is introduced to avoid redundant calculations of LSTM gate operations to reduce system latency. Compared with a software implementation of the AutoEncoder-LSTM on CPU and GPU, our FPGA design is respectively 36.3 and 23.9 times faster and consumes 247 and 499 times less energy.**

I. Introduction

During a test flight of the COMAC (Commercial Aircraft Corporation of China, Limited) C919 airplane, terabytes of high dimension and high frequency data are collected. For flight testing, we need to address three challenges. First, anomaly cases are not easily available. Second, existing techniques based on LSTM are insufficiently accurate in predictions with large dimensional outputs [1]. Third, detection system should always complete operations within a specified time. To address these challenges, this paper proposes a novel approach where LSTM detects anomalies using transformed flight vibration data produced by a fine-tuned autoencoder. This approach minimises the negative impact of the curse of dimensionality as shown in Fig. 1. Furthermore, we propose an FPGA-based architecture with many computation and communication optimisations. A novel Timestep(TS)-buffer is introduced to avoid recalculations of LSTM gate operations to reduce latency. This work is motivated by flight vibration anomaly detection, but applies generally to many other applications using multivariate time series data for anomaly detection on FPGAs.

Although many hardware implementations of LSTM have been reported [2, 3, 4, 5], there are very few reports about AutoEncoder(AE)-LSTM on FPGAs. To the best of our knowledge, we are the first to propose and implement a unified AE-LSTM framework for anomaly detection on FPGAs.

Our contributions in this paper are as follows:

1) An new framework to build anomaly detection models based on AE-LSTM for analysing high-dimensional time series data for flight testing.

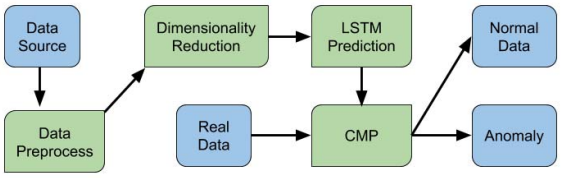

Fig. 1. The Flow of Anomaly Detection System

2) A novel real-time anomaly detection architecture with computation and communication optimisations.
3) Evaluation of the proposed method and hardware architecture.

II. Background and Preliminaries

A. Related Work

A popular machine learning technique for anomaly detection is Support Vector Machine (SVM). FPGA-based SVM architectures have been proposed for anomaly detection in cyber security [6]. Moss et al. [7] present an O(1) time-complexity spectral anomaly detector that achievs a latency of 68 ns. An anomaly detection algorithm for real-time hyperspectral imaging is implemented using FPGA and significantly outperforms the corresponding software version [8]. Furthermore, an FPGA-based autoencoder is proposed for real-time anomaly detection of radio frequency signals in [9]. A hardware architecture for anomaly detection using LSTM has been reported [10], however it cannot handle large dimensions.

B. AutoEncoder & LSTM

An AutoEncoder (AE) is a type of artificial neural network for learning efficient data codings in an unsupervised manner. It learns to transform data from an input layer into a latent-space representation, and then reconstructs the output of the reduced latent representation as close as possible to its original input. This makes the autoencoder suitable for dimensionality reduction when the latent representation has fewer dimensions. Note that in the general autoencoder framework, we can use other forms of functions for the encoder or decoder such as deep fully-connected networks with nonlinearity, Relu CNN or others. In our work, the Batch normalization layer is added in the encoder part to increase the accuracy of the following

978-1-7281-2943-3/19/$31.00 ©2019 IEEE
DOI 10.1109/ICFPT47387.2019.00072

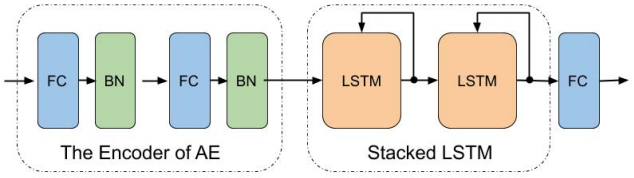

Fig. 2. The Overview of the Neural Network for Anomaly Detection. Only the encoder of AE is included as the decoder part is removed after AE is trained.

LSTM network. After training, the decoder is removed while the encoder is used to generate the reduced representations.

The LSTM architecture relies dedicated memory cells to store information about long-term dependencies, which is well suited for time series data. We follow the implementation of LSTM in [3, 5] and the equations are shown below, where $\odot$ is an element-wise multiplication:

$$\begin{aligned} i_t &= \sigma(W_i[x_t, h_{t-1}] + b_i) \\ f_t &= \sigma(W_f[x_t, h_{t-1}] + b_f) \\ u_t &= \sigma(W_u[x_t, h_{t-1}] + b_u) \\ o_t &= \sigma(W_o[x_t, h_{t-1}] + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot u_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{1}$$

i, f, u and o represent the input, forget, update and output gate respectively. We combine the input vector and hidden vector so that W represents the single weight matrix for both input and hidden elements and b is the bias term. This work will focus on the optimisation of the standard LSTM but the proposed techniques can be applied to other RNN and LSTM variants.

III. Design and Implementation

In this section, an overview of the proposed design is first presented. Then several software and hardware optimisation techniques are proposed.

A. Design Overview

Our proposed approach is shown in Fig. 1. The basic structure of our network is similar to previous LSTM anomaly detector [11], however the input data from flight test vibration sensors are high-dimensional data. So we introduce an autoencoder to reduce the dimensionality of the data. An overview of the neural network used in our anomaly detection system is shown in Fig. 2. Only the encoder of the AutoEncoder is shown in the figure as the LSTM inference procedure does not require data reconstruction. The encoder generates 32-dimensional data using the 1024-dimensional original data. The LSTM-based predictive model makes a prediction for each 32-dimensional data point from the autoencoder. The predictive model comprise two LSTM layers, each containing 64 hidden units. The system flags a data point as an anomaly if the absolute difference between the real and the predicted values is sufficiently high.

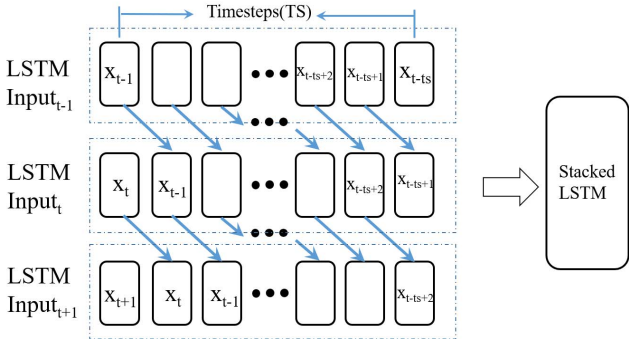

Fig. 3. LSTM Inputs in different timesteps

B. Computation optimisation

There are many redundant calculations of processing input activation (x_t in equations (1)) multiplying LSTM gate weights (W in equations (1)) in each inference when the timestep of LSTM is larger than 1, as shown in Fig 3. Timesteps in LSTM system are the ticks of time. In our system, a time step is associated with one sample vector from the sensors with a sampling rate of 8 kHz. When timestep is larger than 1, the shape of the input is $[timestep, input\_activation]$ and LSTM cell will iterate for $timestep$ times for each inference. However input activations of two adjacent inferences are partially the same so we do not need to re-calculate the matrix-vector (MV) multiplications for the same input activations. For each tick of time (each inference), we calculate the MV multiplications only for a new input vector and store the intermediate results in the Timestep(TS)-buffer which can be fetched later. Thus, in every inference, only one input vector needs multiplications while the results of the other $timestep - 1$ vectors have already been calculated in the former ticks of time so that they can be fetched from the TS-buffer without re-calculations. We only need to calculate the MV multiplications for each hidden units and accumulate the stored results of the input activations in TS-buffer, which means less power and latency. Introducing and developing the TS-buffer carefully do not make our system more complex. There is no complex logic (only one counter) to decide if the inputs are the same. Since the data are time-series and sequential, the first $timestep - 1$ inputs of x in the current inference are the same as the former inference. In addition, in our design, the autoencoder neural network and LSTM neural network share the same MV multiplication kernel to save DSPs usage in FPGA.

C. Software optimisation

In many data processing tasks, the data are of high dimensionality. Without dimensionality reduction, direct analysis usually leads to complex models and intense calculations. Principal component analysis (PCA) is a common method for dimensionality reduction [12, 13, 14], and its implementation process is relatively simple. In our system, we focus AutoEncoder for dimensionality reduction. Furthermore, we

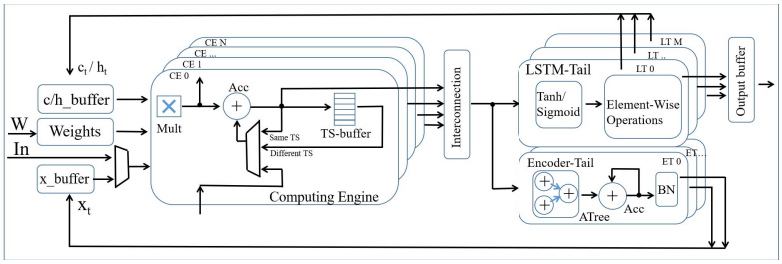

Fig. 4. The Overview of the AE-LSTM Architecture

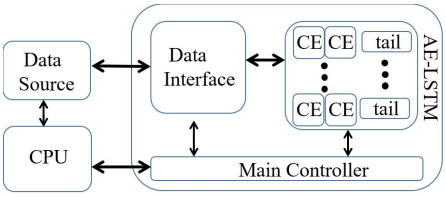

Fig. 5. The Whole System

introduce a Batch Normalization (BN) layer after each fully-connected layer in the encoder part of the autoencoder. From our experiments, it could reduce the Root Mean Square Error (RMSE) of the LSTM training from 0.045 to 0.033 so 36.4% better.

D. Hardware Architecture

Fig. 5 shows an overview of the whole implementation on an FPGA board. It consists of multiple modules that implement the AE-LSTM anomaly detection system. This system consists of the AE-LSTM unit, Data Interface and Main Controller unit. The Main controller unit is used to transfer control commands while data communication is managed by the Data interface unit which is connected to an AXI4 bus, PCIe bus or Ethernet port. When the Data Interface is a normal AXI4 interface with DMA engines, input data come from the external memory, like DDR3 DRAM. The CPU sends configurable commands and input data to the AE-LSTM system and receives the results when the hardware finishes the processing, which is all done via the Main Controller unit.

Based on the optimisation techniques discussed above, the detailed architecture of AE-LSTM is shown in Fig. 4, which is composed of multiple Computing Engines (CEs), a few LSTM-Tail (LT) units and Encoder-Tail (ET) units. In addition, the weights buffer stores all the trained weights for AE and LSTM while the c/h_buffers are used as temporary memories to store the LSTM cell memory/hidden status. Furthermore, the data in x_buffer are the results from the Encoder and also the input vector x of LSTM units. The CEs perform the matrix-vector operations that work as the LSTM gates and FC operations in the Encoder while the TS-buffer is a circular RAM and stores the partial results from input vectors in each time step. If the TS-buffer is full, the oldest entry will be replaced by the new result, which means that the TS-buffer always includes all the required partial results in the current inference and the redundant computations involving the input vectors with the corresponding weights can be avoided. A main Finite-State Machine (FSM) was also used to remove the conflicts, like the conflict between system input and x-buffer input.

TABLE I
RESOURCE UTILIZATION

		LUT	LUTRAM	FF	BRAM	DSP
	Avail.	218600	70400	437200	545	900
CE (256)	Used	29467	12083	34193	133.5	295
	Utili.	13.48%	17.16%	7.82%	24.50%	32.78%

IV. EVALUATION AND ANALYSIS

A. Experimental Setup

In this work, the AE is implemented using Keras and Tensorflow and trained with EarlyStop. After the AE has been trained, the encoder is used to generate the low-dimensional data. Both LSTM layers have 64 hidden units and a timestep of 16. To provide a demonstration, we select 1000 variables from the commercial flight testing vibration data and intercept 10000 lines to train and test our model. The number of the training set is 6416 while the number of validation set is 1600 and the number of the test set is 1984. All data are normalised to 0-1 before they enter the system. To analyse the performance and limitations of the proposed AE-LSTM hardware acceleration, we implement the hardware system using the platform of Xilinx ZC706, which consists of an XC7Z045 FPGA and a dual ARM Cortex-A9 processor.

B. Resource Utilisation

Table I shows the resource utilisation for our AE-LSTM design on the XC7Z045 FPGA. The number of CEs is configured to be 256. The multipliers in the LSTM gates are 16-bit, however the accumulators after the multipliers are 26-bit to retain precision. The multipliers and adders in the LSTM-Tails and BN are both 32-bit.

C. Accuracy and Anomaly Detection

The training and validation losses of LSTM-only, AE-LSTM without BN and AE-LSTM with BN are shown in Fig. 6. AE-LSTM without BN can get a slightly better loss than LSTM-only system. However, the best result is achieved by AE-LSTM with BN. We determine whether the data are abnormal by calculating the difference between the measured value from sensors and the predicted value. The prediction

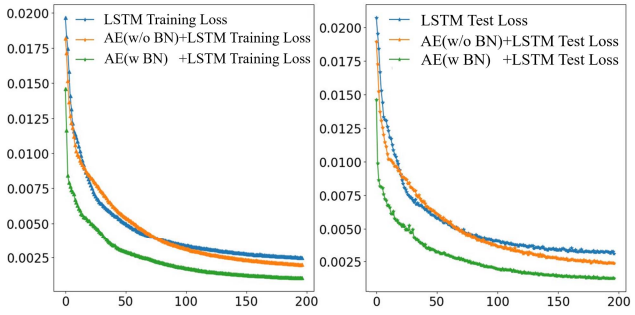

Fig. 6. The training and validation losses of different models

TABLE II
CONFUSION MATRIX

Predict	Acutal	
	Anomaly	Normal
Anomaly	30	11
Normal	0	1927

error vectors are modelled to fit a Gaussian distribution $\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2)$ [11], where $\mathcal{X}$ is a random variable and $\mathcal{N}$ denotes normal distribution with a mean of μ and a variance of σ^2. An observation is predicted as 'anomaly' if its likelihood $p < \tau$ while the τ is set to 0.01 given a sufficient confidence of 99%. 30 synthetic anomalies based on the anomaly patterns provided by the experts of COMAC are introduced in the test dataset and Table II shows the confusion matrix for the AE-LSTM technique. The system detects all the anomalies using AE-LSTM regardless of the presence of dimensionality reduction.

D. Performance and Efficiency Comparison

To compare the performance of the proposed design on FPGA with other platforms, we implement the AE-LSTM system on both Intel Xeon x5690 CPU and Nvidia TITAN GPU based on the Tensorflow framework. Compared with AE-LSTM on CPU and GPU, our FPGA design is 36.3 and 23.9 times faster, 247 times and 499 times more energy efficient respectively as shown in Table III. These significant improvements are due to the low utilisation of GPU [4] which may prefer large matrices without data dependence. Our system achieves 55.53 GOPS which is 4.13 times higher performance than the state-of-the-art FPGA design of LSTM based anomaly detection [10]. However, since detailed accuracy values are not reported in [10], we omit the comparison of accuracy. Furthermore, they do not handle high-dimensional data since they support an LSTM system with a strict bound on data dimension of 19 only.

The sensor sample rate is 8 kHz so the sampling interval of the data is $0.125ms$. With our new hardware architecture and novel neural network, the calculation can finish in $0.043ms$, which can effectively avoid the data accumulation problem.

V. Conclusions and Future Work

This paper proposes an AE-LSTM architecture to accelerate the inference of anomaly detection systems on FPGAs. We have implemented the proposed accelerator on a Xilinx ZC706 FPGA board with excellent performance and efficiency which shows the effectiveness of our approach. Further research includes reducing false positive results, and providing an end-to-end tool chain to automate rapid development of efficient anomaly detection on FPGAs for various applications.

TABLE III
PERFORMANCE COMPARISON OF FPGA DESIGN V.S. CPU AND GPU

	CPU	GPU	Our Work
Platform	Intel Xeon x5690	TITAN X Pascal	Zynq 7Z045
Frequency	3.4 GHz	1.62	142 MHz
Technology	32 nm	16 nm	28 nm
Power(W)	80	60	5.88
Precision	32 bit float	32 bit float	16-32 bit
Time per Sample[a](ms)	1.56	1.03	0.043
Energy per Sample[a](mJ)	124.8	61.8	0.25

[a] Each Sample includes 16 timesteps

ACKNOWLEDGEMENT

The support of the United Kingdom EPSRC (grant numbers EP/L016796/1, EP/N031768/1, EP/P010040/1 and EP/L00058X/1), Natural Science Foundation (No. 61201059) of China, the Strategic Priority Research Program of Chinese Academy of Sciences (grant No. XDA19000000) and Corerain Technologies is gratefully acknowledged.

References

[1] K. Hundman *et al.*, "Detecting spacecraft anomalies using LSTM and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[2] V. Rybalkin *et al.*, "FINN-L: Library extensions and design trade-off analysis for variable precision LSTM networks on FPGAs," in *28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018.

[3] Y. Guan *et al.*, "FPGA-based accelerator for long short-term memory recurrent neural networks," in *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017.

[4] J. Fowers *et al.*, "A Configurable Cloud-Scale DNN Processor for Real-Time AI," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, 2018.

[5] Z. Que *et al.*, "Efficient weight reuse for large lstms," in *30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2019.

[6] A. Bara, X. Niu, and W. Luk, "A dataflow system for anomaly detection and analysis," in *International Conference on Field-Programmable Technology (FPT)*. IEEE, 2014, pp. 276–279.

[7] D. J. Moss *et al.*, "An FPGA-based spectral anomaly detection system," in *International Conference on Field-Programmable Technology (FPT)*. IEEE, 2014.

[8] B. Yang *et al.*, "Dual-mode FPGA implementation of target and anomaly detection algorithms for real-time hyperspectral imaging," *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2015.

[9] D. J. Moss *et al.*, "Real-time FPGA-based anomaly detection for radio frequency signals," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018.

[10] Z. Sun *et al.*, "Fpga acceleration of lstm based on data for test flight," in *IEEE International Conference on Smart Cloud (SmartCloud)*, 2018.

[11] P. Malhotra *et al.*, "Long short term memory networks for anomaly detection in time series," in *Proceedings*. Presses universitaires de Louvain, 2015.

[12] A. Das *et al.*, "An FPGA-based network intrusion detection architecture," *IEEE Transactions on Information Forensics and Security*, 2008.

[13] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[14] S. T. Roweis, "EM algorithms for PCA and SPCA," in *Advances in neural information processing systems*, 1998, pp. 626–632.