Fixed-Point Evaluation of Extreme Learning Machine for Classification

Yingnan Xu, Jingfei Jiang, Juping Jiang, Zhiqiang Liu and Jinwei Xu

Abstract With growth of data sets, the efficiency of Extreme Learning Machine (ELM) model combined with accustomed hardware implementation such as Field-programmable gate array (FPGA) became attractive for many real-time learning tasks. In order to reduce resource occupation in eventual trained model on FPGA, it is more efficient to store fixed-point data rather than double-floating data in the on-chip RAMs. This paper conducts the fixed-point evaluation of ELM for classification. We converted the ELM algorithm into a fixed-point version by changing the operation type, approximating the complex function and blocking the large-scale matrixes, according to the architecture ELM would be implemented on FPGA. The performance of classification with single bit-width and mixed bit-width were evaluated respectively. Experimental results show that the fixed-point representation used on ELM does work for some application, while the performance could be better if we adopt mixed bit-width.

Keywords Extreme Learning Machine (ELM) · Fixed-point evaluation · Classification

1 Introduction

Due to advances in technology, the size and dimensionality of data sets used in machine learning tasks have grown very large and continue to grow by the day. For this reason, it is important to have efficient computational methods and algorithms that can be applied on very large data sets, such that it is still possible to complete the machine learning tasks in reasonable time [8].

Extreme Learning Machine (ELM) is well known for its computational efficiency, making it well-suited for large data processing. However, it is still worth speeding

Y. Xu (✉) · J. Jiang · J. Jiang · Z. Liu · J. Xu
Science and Technology on Parallel and Distributed Processing Laboratory,
National University of Defense Technology, ChangSha 410073, Hunan, China
e-mail: yingnanpearl@163.com

J. Cao et al. (eds.), *Proceedings of ELM-2015 Volume 1*,
Proceedings in Adaptation, Learning and Optimization 6,
DOI 10.1007/978-3-319-28397-5_3

up its implementation in many real-time learning tasks. Hardware implementation is one of the most popular approaches, and two types of reconfigurable digital hardware have been adopted, i.e., Field-programmable gate array device (FPGA) and complex programmable logic device (CPLD) [3]. As rival application-specific integrated circuit, FPGA can attain performances and logic densities at lower development costs and privilege computational optimization over area optimization, thus many of previous works [6] implementing other machine learning algorithms have applied FPGA. The parameters, such as weights and Bias of hidden nodes, are stored in on-chip RAM during processing and are swapped out to off-chip memory after processing. Since it is too expensive to support a large number of floating-point units on chip and store values using the standard double precision floating-point representations in on-chip RAMs, most of these works has adopted fixed-point data. Bit-widths with integral multiple of bytes are convenient to align with other components (such as IP cores and user interfaces) and easier to design. Recent work [2] found that very low precision storage is sufficient not just for running trained networks but also for training them by training the Maxout networks with three distinct storing formats: floating point, fixed point and dynamic fixed point. However, the efficiency of ELM model combined with FPGA adopting fixed bit-width is not very clearly.

Motivated by this gap between workloads and state-of-the-art computing platforms, we evaluate fixed-point implementation of ELM for classification. At first, we present the architecture of FPGA. Then with this in mind, we converted the ELM algorithm into a fixed-point version by changing the operator type, approximating the complex function and blocking the large-scale matrixes. Finally, we evaluated the performance of classification with single bit-width and mixed bit-width respectively.

Experiments are performed on a large data setSatImage. Results of the experiments show it does work for some application to use the fixed-point representation on ELM, however, considering resource occupation, the performance of implementation adopting single bit-width is not so optimistic, and it could be improved if we adopt mixed bit-width.

The organization of this paper is as follows. Section 2 introduces the algorithm of ELM. Section 3 shows the specific procedure of fixed-point conversion for ELM including changing the operator type, approximating the complex function and blocking the large-scale matrixes. Section 4 presents our experiment and the results of simulations adopting single bit-width and mixed bit-width respectively. Finally, the results are discussed and an overview of the work in progress is given.

2 Extreme Learning Machine (ELM)

ELM was proposed for generalized single-hidden layer feedforward networks where the hidden layer need not be neuron alike. It offers three main advantages: low training complexity, the minimization of a convex cost that avoids the presence of

local minima, and notable representation ability. The output function of ELM for generalized SLFNs is

$$\mathrm{f_L}\,(\mathrm{x}) = \sum_{\mathrm{i}=1}^{\mathrm{L}} \beta_i \mathrm{h}_i\,(\mathrm{x}) = \mathrm{h}\,(\mathrm{x})\,\beta \quad , \tag{1}$$

where $\beta = \left[\beta_1, \ldots, \ \beta_{\mathrm{L}}\right]^{\mathrm{T}}$, is the output weight vector between the hidden layer of L nodes to the $m1$ output nodes, and $h(x) = [h_1(x), \ldots, h_L(x)]$ is ELM nonlinear feature mapping, $h_i(x)$ is the output of the ith hidden node output. In particular, in real applications $h_i(x)$ can be

$$h_i(x) = G(a_i, b_i, x), a_i \in R^d, b_i \in R \ . \tag{2}$$

Basically, ELM trains an SLFN in two main stages: (1) random feature mapping and (2) linear parameters solving. In the first stage, ELM randomly initializes the hidden node parameters (a, b) to map the input data into a feature space by nonlinear piecewise continuous activation function. The most often used activation function is sigmoid function, the formula is

$$\mathrm{G}\,(\mathrm{a}, \mathrm{b}, \mathrm{x}) = \frac{1}{1 + \exp\left(-\left(ax + b\right)\right)} \ . \tag{3}$$

In the second stage of ELM learning, the weights connecting the hidden layer and the output layer, denoted by β, are solved by minimizing the approximation error in the squared error sense:

$$\min_{\beta \in R^{L \times m}} \|H\beta - T\|^2 \ , \tag{4}$$

where H is the hidden layer output matrix (randomized matrix):

$$\mathrm{H} = \begin{bmatrix} h\left(x_1\right) \\ \vdots \\ h\left(x_N\right) \end{bmatrix} = \begin{bmatrix} h_1\left(x_1\right) & \cdots & h_L\left(x_1\right) \\ \vdots & \vdots & \vdots \\ h_1\left(x_N\right) & \cdots & h_L\left(x_N\right) \end{bmatrix} , \tag{5}$$

and T is the training data target matrix:

$$\mathrm{T} = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \vdots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix} , \tag{6}$$

where $\|\cdot\|$ denotes the Frobenius norm.

The optimal solution to (4) is given by

$$\beta^* = H^{\dagger} T \ , \tag{7}$$

where $H^{\dagger}$ denotes the MoorePenrose generalized inverse of matrix H.

A positive value C can be added to the diagonal of H^TH or HH^T of the Moore-Penrose generalized inverse H the resultant solution is more stable and tends to have better generalization performance [4]. Thus

$$\beta^* = (H^TH + 1/C)^{-1}H^TT\ , \tag{8}$$

where I is an identity matrix of dimension L.

Overall, the ELM algorithm is then:

ELM Algorithm: Given a training set $\aleph = \{(x_i, t_i)\,|x_i \in R^n, t_i \in R^m, i = 1, \ldots, N\}$, hidden node output function G(a, b, x), and the number of hidden nodes L,

1. generate random hidden nodes (random hidden node parameters) $(a_i, b_i), i = 1, \ldots, L$.
2. Calculate the hidden layer output matrix H.
3. Calculate the output weight vector $\beta^* = (H^TH + 1/C)^{-1}H^TT$.

3 Fixed-Point Conversion for ELM

Since FPGAs attain performances and logic densities at lower development costs and the resource consumption can be further reduced with fixed-point data, we attempt to implement the ELM algorithm for classification on FPGA using fixed-point format, the overall FPGA architecture of ELM algorithm for classification is shown in Fig. 1. According to previous works [7], the QR decomposition adopts float-point format while the multiplication of matrix adopts fixed-point format. In this work, we simulate the behaviour of FPGA adopting fixed bit-width on Matlab environment. Figure 2 shows the execution flow.

In the beginning of simulation, we adjust the radix point position shown in Fig. 3 [6] according to the corresponding range of data. For example, since the range of InputWeight matrix shown in Fig. 2 is approximately $[-1, 1]$, the optimal bit-width of integer part is 1 and cannot be allowed any further reduction or increment. This way can make the precision better when considering a fixed-point representation for real numbers, the integer part of a number mainly influences the representation scope while the fractional part mainly decides the precision.

In the procedure of fixed-point conversion, we choose Piecewise Linear Approximation of nonlinearity algorithm (PLA) [1] as the method of sigmoid function approximation, this method uses linear functions and can be implemented on FPGA easily. PLA has uniform structures like Table 1. For the main operations like matrix multiplication in ELM, parallel multiply-accumulators are often used on FPGA. The operands are stored on distributed block RAM, which bit-width is n bits. A 2n bits partial product can be produced by the n bits multiplier. An accumulator with larger bit-width can be used to accumulate the partial product, avoiding the precision lost and not increasing much logic cost at the same time. So, we often chose a bit-width

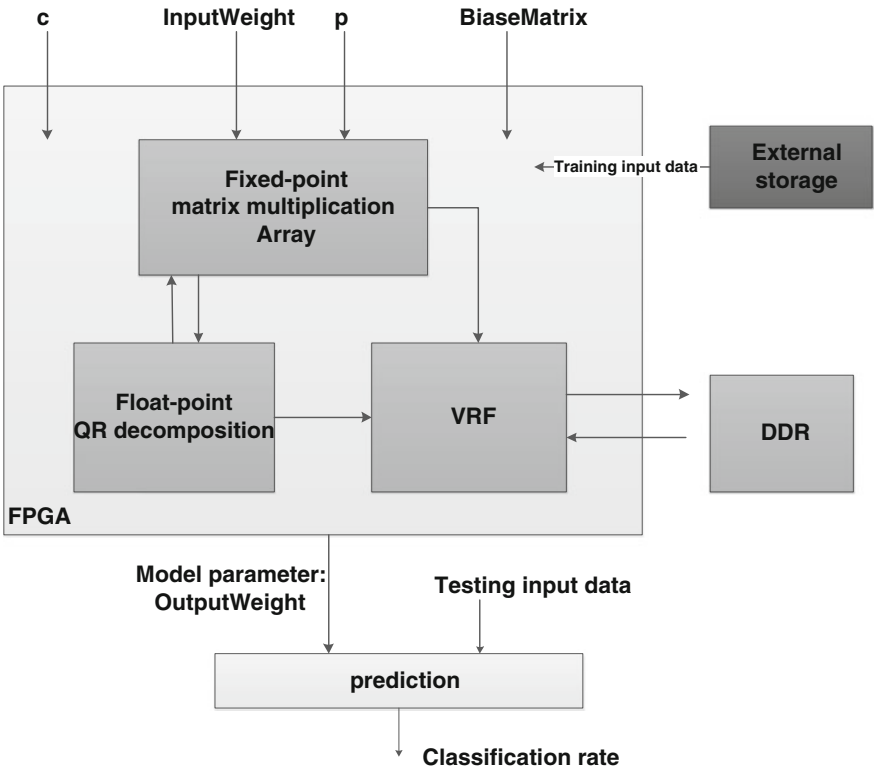

Fig. 1 FPGA architecture of ELM algorithm for classification

in the range of n bits to 2n bits for the adder and the accumulator. Only the bit-width of the final result which needs to store back to on-chip RAM is constrained to n bits. The partition of the integer part and fractional part for the result depends on the representation range of the data, which must be studied when converting to the fixed-point hardware.

Under the implementation assumption above, it is more reasonable that maintaining the precision of a block matrix multiplication instead of converting the partial product for each element. Assuming that we can chose enough wide bit-width for the accumulation operation, thus we only need to cut down the bit-width to n bits for the result of a block multiplication when simulating the fixed-point operations. From this observation, we converted all matrix operations in ELM to a loop code of block matrix operations and converted each element of the block result to a fixed-point format. The size of block matrix is set 64. The flow diagram of computation is shown in Fig. 4.

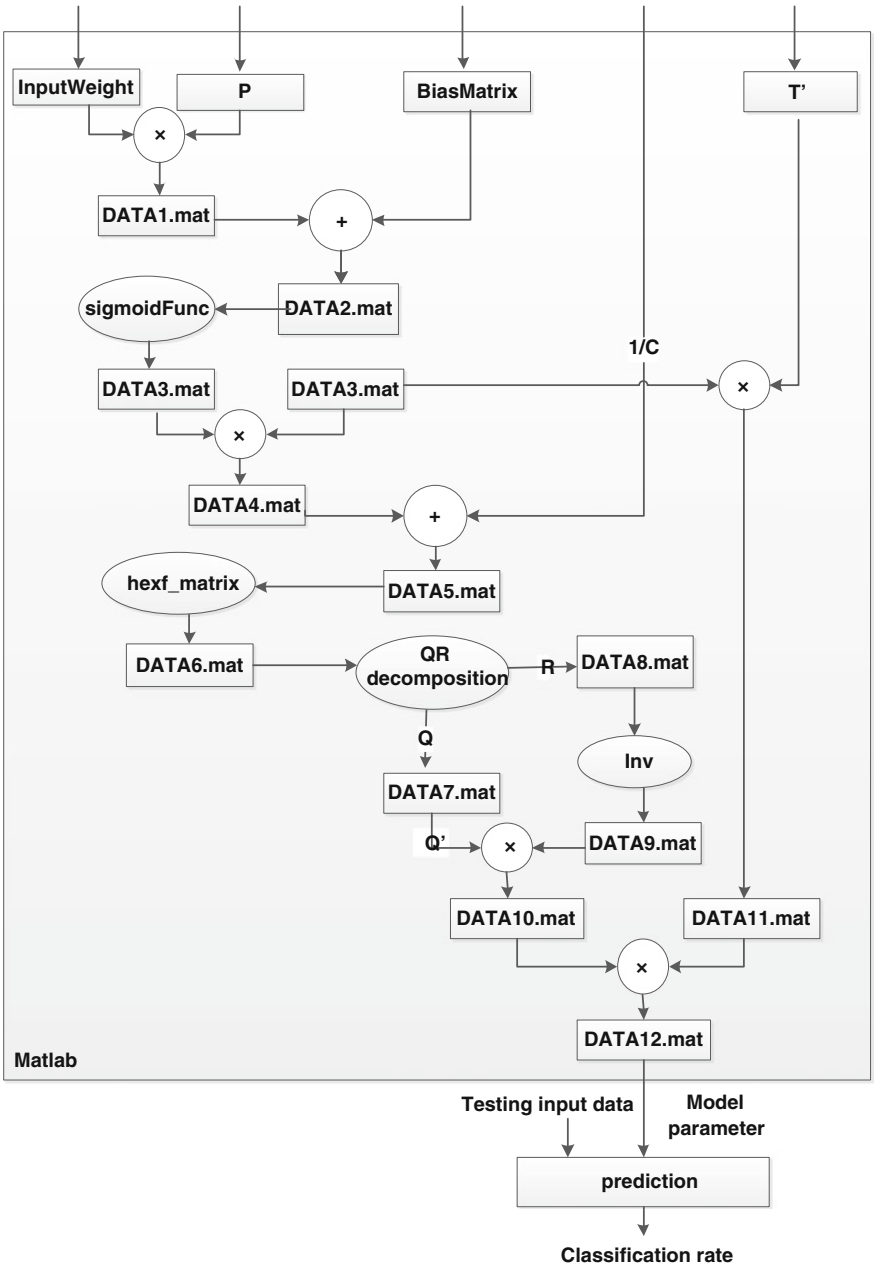

Fig. 2 Execution flow of ELM for classification

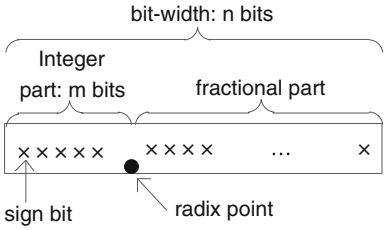

Fig. 3 Fixed-point data format

Table 1 Piecewise linear approximation algorithm

x	y
$0 \le \lvert x\rvert < 1$	$y = (\lvert x\rvert + 2)/4$
$1 \le \lvert x\rvert < 19/8$	$y = (\lvert x\rvert + 5)/8$
$19/8 \le \lvert x\rvert < 5$	$y = (\lvert x\rvert + 27)/32$
$\lvert x\rvert \ge 5$	$y = 1$
$x < 0$	$y = 1 - y$

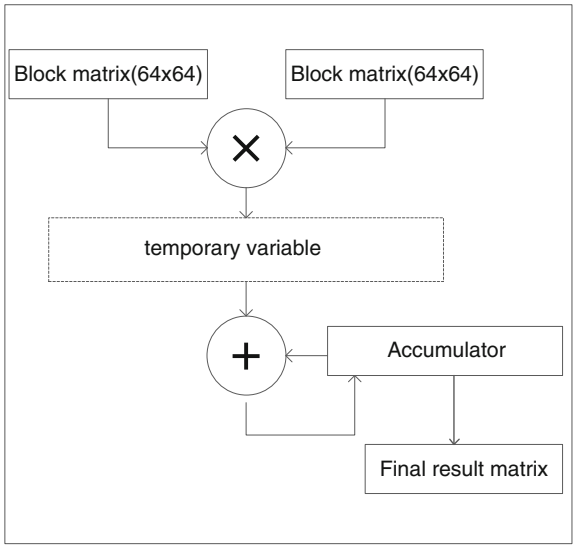

Fig. 4 Flow diagram of block matrix multiplication

4 Experiment and Results

All the simulations are conducted in MATLAB R2009a environment on an ordinary PC with Intel(R) Core(TM) i3-2120 and 4 GB RAM.

The SatImage dataset with 36 input attributes and 6 class label is chosen as experimental Dataset. It can be downloaded from the official ELM website with pre-scaled values [6]. 3217 instances are used as training data and the rest 3218 of the instances are used for testing.

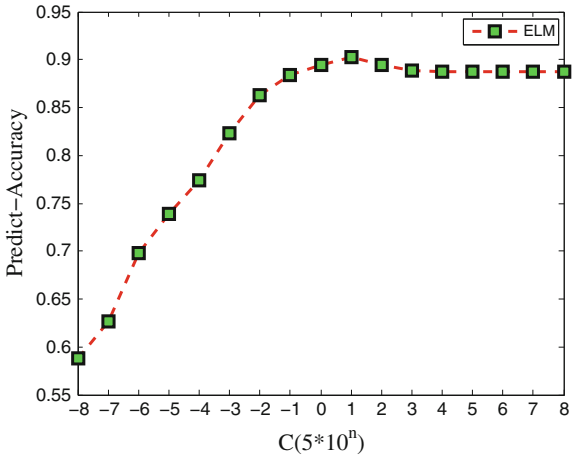

Fig. 5 Classification accuracy with different C

In ELM, the number of output nodes is equal to the number of classes. For the SatImage data set used in this paper, there are 6 classes and, thus, ELM has 6 output nodes. The activation function used in ELM in our experiment is sigmoid function.

In the implementation of ELM, it is found that the generalization performance of ELM is not sensitive to the dimensionality of the feature space (L) and good performance can be reached as long as L is large enough. In our simulations, $L = 1000$ is set for all tested cases no matter whatever size of the training data sets. And since training data sets are very large $N \gg L$, we apply solutions (11) in Sect. 2 to reduce computational costs [5].

The hidden node parameters a_i and b_i are not only independent of the training data but also of each other. Unlike conventional learning methods which MUST see the training data before generating the hidden node parameters, ELM could generate the hidden node parameters before seeing the training data. Thus, a set of random values are produced to be applied in all of our experiment.

And the value C can affect the performance to a large extent. In our experiment, we first trained the ELM classification problem with different C in float-point algorithm. And the Fig. 5 shows the classification accuracy with different value C. It can be seen that the accuracy with C being set 50 is much better than other chosen value. And in order to make C expressed in fixed-point algorithm more accurate, we ignored the possibility that the absolute value of C can be set too large. So, in the following experiment, we applied the fixed value with $C = 50$.

4.1 Single Bit-Width

The Fig. 6 shows the classification accuracy applying different fixed bit-width. It can be seen that the accuracies with the bit-width set 16 bits, 24 bits and 32 bits are all bad, however the performance in the bit-width of 16 bits is better than 24

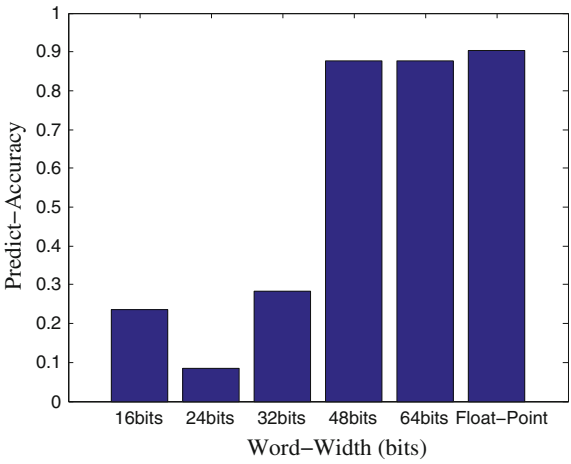

Fig. 6 Prediction accuracy with different bit-widths

bits, it means that the representation domain constraint throws away the redundant and useless information of the high-dimensional input data. The performance in the bit-widths of 48 bits and 64 bits indicate that fixed-point representation used on ELM does work for some application. In order to balance classification accuracy and resource occupation in the eventual trained model, we can only choose 48 bits as the optimal bit-width on FPGAs if we adopt single bit-width. It is obvious that the result is not optimistic.

To solve this problem, we analyzed the result of each operation and tracked the source where error comes from. In this subsection, we computed the Forbenius Norms (FN)

$$\|A - B\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left|a_{ij} - \mathrm{b}_{ij}\right|^2} \tag{9}$$

of error matrixes which can weigh the degree of error, the error matrixes are subtraction between float-point and fixed-point data from the output generated by each execution stage shown in Fig. 2, the result of computation is presented in Fig. 7. It can be seen that the error mainly begins with the operation of large scale matrix multiplication generating DATA4.mat and is propagated in latter operations. Because of the linear nature of the operations and the dynamic range compression of the sigmoid generating DATA7.mat, quantization errors tend to propagate sub-linearly and not cause numerical instability [9].

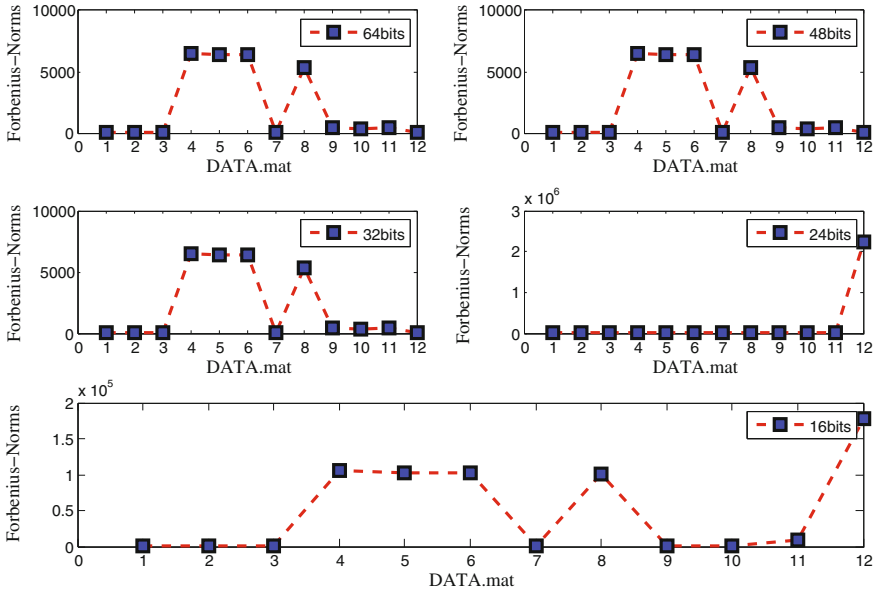

Fig. 7 FN with different bit-widths

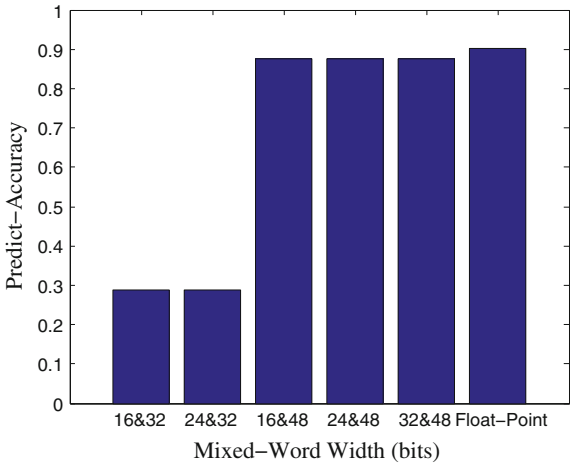

Fig. 8 Prediction accuracy with different mixed bit-widths at DATA4.mat

4.2 Mixed Bit-Widths

In order to improve the performance, we re-trained the ELM by adopting mixed bit-widths which can change bit-width at a special point. The prediction accuracy of training with mixed bit-widths applied at the point computing DATA4.mat is shown in Fig. 8, it can be seen that we can also get attractive result even we adopt mixed

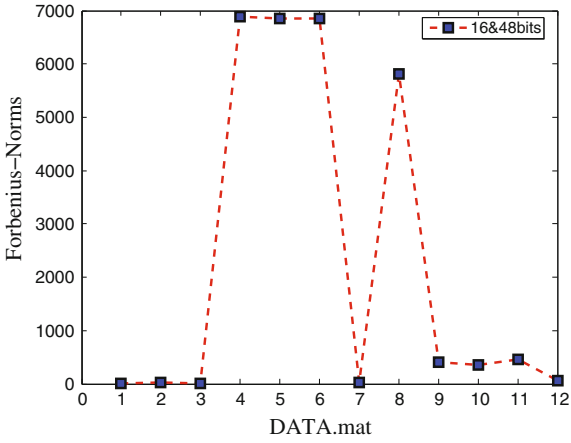

Fig. 9 FN with mixed bit-widths 16&48

bit-widths which can decrease the occupation of memory resource. According to the FN of the optimal mixed bit-widths (16&48) shown in Fig. 9, the propagated error produced by bit-width of 16 bits can be improved through changing the bit-width into 48 bits and would not affect the performance.

5 Conclusion

This research has tackled the fixed-point evaluation of ELM for classification. We conduct the conversion of fixed-point for ELM and then make simulations on Matlab. Experimental results show that the resource occupation of implementation adopting single bit-width is too large, and the performance could be improved if we adopt mixed bit-width. Our results can act as a guide to inform the design choices on bit-widths when implementing ELM in FPGA documenting clearly the trade-off in accuracy. However, the use of mixed bit-width makes the computing resource rise, we need to conduct the further evaluation of resource occupation and then implement the ELM for classification on FPGA with the parameter discussed in this work.

Acknowledgments This work is funded by National Science Foundation of China(number 61303070).

References

1. Amin, H., Curtis, K., Hayes Gill, B.: Piecewise linear approximation applied to nonlinear function of a neural network. IEE Proc. Circuits Devices Syst. **144**(6), 313–317 (1997)
2. Courbariaux, M. Bengio, Y. David, J.P.: Low precision storage for deep learning. C. Eprint Arxiv (2014)

3. Decherchi, S., Gastaldo, P., Leoncini, A., et al.: Efficient digital implementation of extreme learning machines for classification. IEEE Trans. J. Circuits Syst. II: Express. Briefs **59**(8), 496–500 (2012)
4. Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. J. Technometrics **12**(1), 55–67 (2012)
5. Huang, G.B., Zhou, H., Ding, X., et al.: Identification of common molecular subsequences. IEEE Trans. J. Syst. Man Cybern. Part B: Cybern. **42**(2), 513–529 (2012)
6. Jiang, J., Hu, R., Zhang, F., et al.: Experimental demonstration of the fixed-point sparse coding performance. J. Cybern. Inform. Technol. **14**(5), 40–50 (2014)
7. Jie, Z.: Fine-grained algorithm and architecture for data processing in SAR applications (in Chinese). University of Defense and Technology (2010)
8. van Heeswijk, M., Miche, Y., Oja, E., et al.: GPU-accelerated and parallelized ELM ensembles for large-scale regression. J. Neurocomput. **74**(16), 2430–2437 (2011)
9. Vanhoucke, V., Senior, A., Mao, M.Z.: Improving the speed of neural networks on CPUs. In: Proceedings of Deep Learning and Unsupervised Feature Learning Workshop Nips (2011)