CRC: Compressed Reservoir Computing on FPGA via Joint HSIC LASSO-based Pruning and Quantization

Atousa Jafari[1], Hassan Ghasemzadeh Mohammadi[2], and Marco Platzner[1]
[1]*Department of Computer Engineering,* Paderborn University, Germany
[2]*Reneo Group GmbH,* Hamburg, Germany
Email: [1]{atousa.jafari , platzner}@uni-paderborn.de,[2] ghasemzadeh@reno.de

***Abstract*—While reservoir computing (RC) networks offer advantages over traditional recurrent neural net- works in terms of training time and operational cost for time-series applications, deploying them on edge devices still presents significant challenges due to re- source constraints. Network compression, i.e., pruning and quantization, are thus of utmost importance. We propose a Compressed Reservoir Computing (CRC) framework that integrates advanced pruning and quantization techniques to optimize throughput, latency, energy efficiency, and resource utilization for FPGA-based RC accelerators.**

We describe the framework with a focus on HSIC LASSO as a novel pruning method that can capture non-linear dependencies between neurons. We validate our framework with time series classification and regression tasks, for which we generate FPGA accelerators. The accelerators achieve a very high throughput of up to 188 Megasamples/s with a latency of 5.32 ns, while reducing resource utilization by 12× and lowering the energy by 10× compared to a baseline hardware implementation, without compromising accuracy

***Keywords*—Dataflow accelerator, Echo state network, Pruning, Quantization, Time-series application**.

I. Introduction and Background

Reservoir computing (RC) has emerged as a promising alternative to traditional recurrent neural network (RNNs), offering a simpler and more efficient approach to time-series analysis. The most common variant of RC is the Echo State Network (ESN), which consists of an input layer, a reservoir layer, and an output layer as illustrated in Figure 1. The input layer is connected to the neurons in the reservoir layer through randomly generated synaptic connections with weights, modeled as a matrix $\boldsymbol{W_{in}}$. The reservoir layer contains neurons with randomly initialized sparse interconnections, represented by the matrix $\boldsymbol{W_r}$. The reservoir is the core of an ESN, where the feedback connections of neurons together with the non-linearity of their activation functions form a high- dimensional dynamical and non-linear system. The output layer is connected to the reservoir via weighted connections, denoted as $\boldsymbol{W_{out}}$. Unlike standard RNNs, where all layers are trained, an ESN simplifies the training process by randomly initializing and fixing the input and reservoir layers. Only the output layer is trained using basic regression techniques, significantly reducing the computational cost and complexity of training [1]. The reduced effort for training and the rather simple layer structure make RC approaches well suited for time-series tasks on edge devices, particularly for non-linear time series forecasting and time-series classification [2]. This increased network size results in significant computational effort and energy requirements during inference. Thus, effective network compression techniques are studied to reduce the network size and the computational load without compromising performance [3], [4].

For example, network pruning techniques to reduce the number of neurons and connections are discussed in [4]–[6], and quantization for RC models is studied in [2], [7], [8]. In our previous work [9], we presented an approach to map ESN to FPGA hardware as a fully unrolled and quantized dataflow streaming architecture that achieves ultra-low latency and extreme throughput.

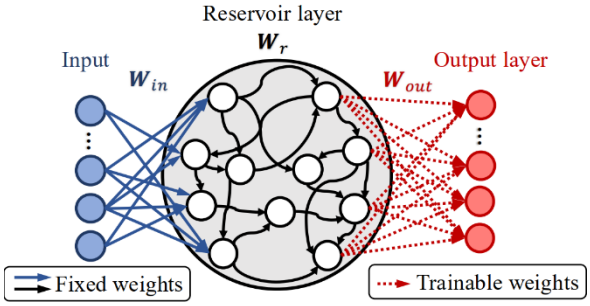


Fig. 1. Layer model of echo state networks (ESN).

In this work-in-progress paper, we propose the feature selection technique HSIC LASSO to be used as a novel pruning algorithm for RC models. HSIC LASSO identifies and removes less important neurons by considering nonlinear correlations within the network, which is a novelty over related work. We present a Compressed Reservoir Computing (CRC) framework for the efficient mapping of RC models to FPGAs, combining pruning and quantization as compression techniques. We experimentally study the effects of pruning and quantization and show that we can reduce the hardware resource requirements up to 12× and decrease the energy by 10× compared to a 32-bit fixed- point baseline hardware implementation.

II. Proposed CRC Framework

Figure 2 depicts the overall flow of our proposed framework for Compressed Reservoir Computing (CRC) on FPGAs. The flow includes four main steps. The first step is *Network Initialization*, where we construct and train an RC model for a given dataset. This step leverages the ReservoirPy framework [10] and includes hyperparameter optimization to achieve the required accuracy. The second step is *HSIC LASSO Pruning*, where we eliminate less significant neurons from the model by considering their non-linear correlations. The third step is *Quantization and Streamlining*, which uses a hardware-friendly streamlining approach to quantize all layers of the RC using the Brevitas framework [11]. The final step is *Direct Logic Implementation*, where we convert the compressed RC model into an FPGA design by mapping all RC layers onto LUT-based structures and creating an RTL (Register Transfer Level) description for the overall design. Subsequently, we synthesize the design into hardware using Xilinx Vivado and evaluate parameters such as the finally achieved accuracy, hardware resource usage, throughput, latency, and power. The remainder of this section details the novel HSIC LASSO-based pruning for RC, followed by an

Manuscript received May 19, 2025; revised July 30, 2025; accepted August 13, 2025. Published September 2, 2025.
Issue category: Special Issue on DSD/SEAA 2025 on Works in Progress (WiP) Session, Salerno, Italy, Sept. 2025
Paper category: Short
DOI: doi.org/10.64552/wipiec.v11i1.99

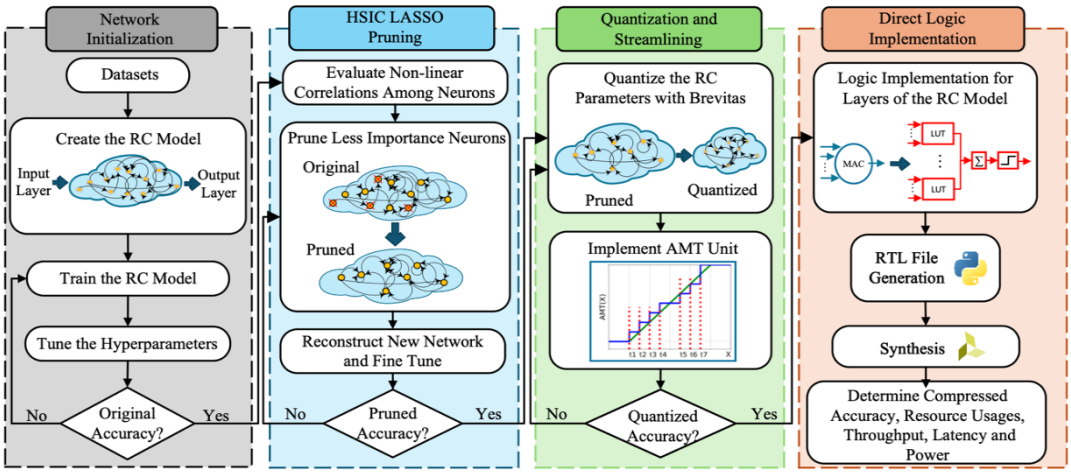


Fig. 2. Overview of our proposed CRC framework for FPGA-based implementation.

overview of the streamlining quantization approach. The last two steps of our flow were elaborated in more detail in [9].

A. *Pruning Echo State Networks via HSIC LASSO*

Traditional methods to identify and prune less contributing neurons in an ESN include Spearman [12], PCA [13] and LASSO [13]. Spearman directly assesses a neuron's contribution to accuracy by measuring how well its activity predicts the final output. Keeping a neuron with high correlation helps minimize the prediction error. PCA work indirectly as it ranks neurons based on their contribution to the reservoir's internal dynamic "richness," not to the final output. Hence, PCA provides a superior feature space for the output layer to learn from, thereby reducing the error. LASSO directly ranks neurons by their importance in minimizing prediction errors within a simplified linear model. By forcing the weights of nonessential neurons to zero, it explicitly identifies and removes the neurons that can be ignored with the least impact on accuracy. All this methods are iterative removing neuron by neuron and retraining the network as long as the desired accuracy is retained.

None of the traditional methods, however, captures the nonlinear correlation that exist between the neurons of the reservoir and the reservoir and the neurons of the output layer. HSIC LASSO leverages the Hilbert-Schmidt Independence Criterion (HSIC) to measure nonlinear dependencies, making it particularly suitable for pruning ESNs. In contrast to traditional methods, it provides an efficient alternative by enabling the selection and removal of redundant neurons in a single step, once the hyperparameters are selected. After removal of redundant neurons, the network is retrained.

HSIC LASSO [14] is an extension of the traditional LASSO method, which replaces the mean squared error component of the objective function with a nonlinear dependency measure. The objective function of HSIC LASSO is formulated as:

$$argmin\,(\tfrac{1}{2}\ ||R_C - \textstyle\sum_{i=1}^{n} W_i U_c^{(i)}\ ||_F^2 A = \pi r^2 + \lambda||w||_1)$$

$$subject\ to\ w_i \geq 0\ \forall i \qquad (1)$$

Where $|.|_F$ denotes the Frobenius norm, and $w \in R^n$ in is the weight vector that determines the contribution of each neuron. The matrices $R_c \in R^{d\times d}$ and $U_c^{(i)} \in R^{d\times d}$ are centered Gram matrices of $R_{j,k} = R(y_{i,j}, y_{i,k})$ and $U_{j,k}^{(i)} = U(x_{i,j}, x_{i,k})$, respectively. In this context, the Gram matrix R captures the pairwise similarities between the states of neurons in the reservoir layer. Each element $U_{j,k}^{(i)}$ is computed using a kernel function $U(.,.)$, such as the Gaussian kernel, which measures the similarity between the states of the j-th and k-th neurons at time step i. Similarly, the Gram matrix R captures the pairwise similarities between the network outputs.

B. *Quantization Based on Streamlining Approach.*

We introduce a hardware-friendly quantization approach with the so-called streamline deployment for quantized RC networks. In this method, floating point (FP) operations (e.g., scale and bias parameters) extracted from quantization are absorbed into the activation function for an efficient hardware implementation according to streamline algorithm described in [9], [15]. The quantized activation layer ($HardTanh$ ($Q_{HardTanh}$) is converted into a successive multi-threshold (MT) layer by dividing the range of the activation function into $2^K - 1$ discrete levels, where K is the bit-width of the quantized activation function. The difference between these levels, referred to as the $step$ corresponds to the $Scale$ of the activation function. The input is then compared to these threshold values, and the closest threshold index is selected as the nearest integer. However, the floating-point $Scale$ persists in the process. To eliminate this as well, we divide each threshold by the $Scale$, round it to the nearest integer, and then multiply by the $step$. This method, called absorbing multi-threshold (AMT), absorbs all FP calculations into the successive multi-threshold process.

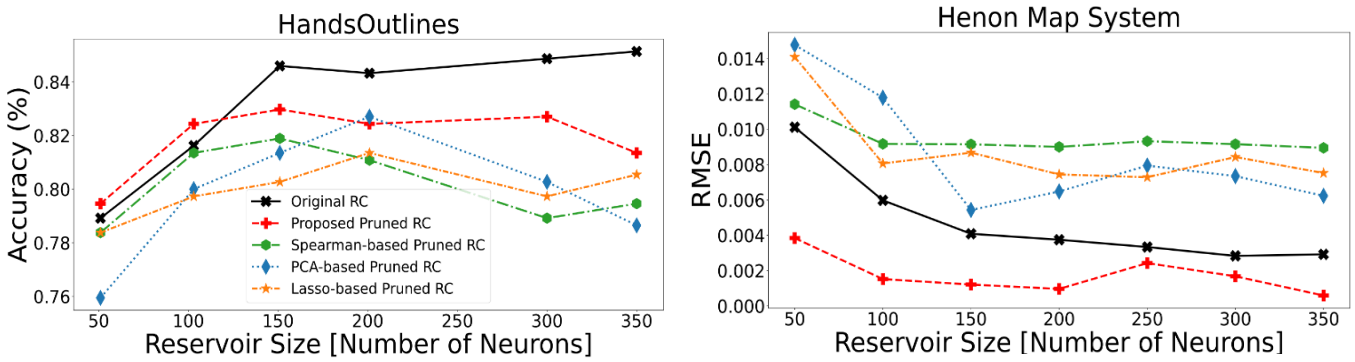


Fig.3. Performance (accuracy, RSME) of the original and pruned networks across reservoir sizes.

To implement the streamlined approach in our framework, we replace the original state update and output equations presented in literature [1] with the updated equations given in Eq. 1 and Eq. 2. In these modified equations, $Q_{u(t)}$, Q_{Win}, $Q_{x(t)}$, Q_{Wr}, and Q_{Wout} denote the integer forms of quantized input, input weight, state and reservoir weight, and output weight, respectively.

$$x(t) = \left(AMT_{Input}\left(Q_{Win} \times Q_{u(t)}\right) + AMT_{Reservoir}\left(Q_{x(t-1)} \times Q_{Wr}\right)\right) \times step \quad (1)$$

$$y(t) = Q_{Wout} \times Q_{x(t)} \times S_{final)} \quad (2)$$

TABEL I: Hardware results for the CRC framework on FPGA (worst-case regression/classification, after pruning).

Accelerator	Network Size	Bit-width (K)	LUTs	FFs	Throughput [Msps]	Latency [ns]	PDP [µWs]
Baseline RC	N_{RC}= 200	32-bit fixed-point	1,629.2K	923.7K	120	8.34	3.31
Quantized RC	N_{RC} = 200	8-bit quantized	21.9K	14.0K	185	5.40	0.65
Pruned RC	N'_{RC} = 149	32-bit fixed-point	228.8K	134.5K	125	7.97	2.34
Compressed RC	N'_{RC} = 149	8-bit quantized	4.6K	2.7K	188	5.32	0.31

TABEL II: PERFORMANCE OF ORIGINAL VS. COMPRESSED 200-NEURON RC MODELS WITH COMPRESSION RATIO

HandsOutlines: Original Accuracy = 83.78%, HSIC LASSO-based Pruned Accuracy = 81.08%, Compression Ratio = **1.8**×
Henon Map System: Original RMSE = 0.003, HSIC LASSO-based Pruned RMSE = 0.002, Compression Ratio = **3.8**×

III. EXPERIMENTAL EVALUATION

A. EVALUATION OF CRC FRAMEWORKE

We evaluate our framework on two widely used RC benchmarks: The HandsOutlines dataset as a time-series classification task, and the Henon Map dataset as a regression task for time-series forecasting. Figure3 presents a comprehensive comparative analysis of the ESN model performance, measured as accuracy for HandsOutlines and RMSE (root-mean square error) for Henon Map for the original un-pruned model, models pruned with Spearman, PCA, LASSO, and with our proposed HSIC LASSO-based technique for varying reservoir sizes.

To get a meaningful and fair comparison, we have varied the reservoir size from 50 to 350 and determined the pruned network sizes using the HSIC LASSO technique. Then, we have used the related pruning techniques to reduce the networks to the same sizes. Hence, we compare the pruning techniques at the same compression ratios. The compression ratio is defined in Eq.3, where N_{RC} and N'_{RC} depict the reservoir size in the numbers of neurons for the original and pruned networks.

$$Compression\ Ratio = \frac{N_{RC}}{N'_{RC}} \quad (3)$$

Figure [3] shows that HSIC LASSO performs better than related pruning methods for both classification and regression tasks, the exception being PCA which achieves a slightly better accuracy for HandOutlines with reservoir size 200. For the regression task, our proposed pruning method even achieves lower error than the original un-pruned model, which points to overfitting. As depicted in Table II, HSIC-LASSO can reduce the number of neurons by almost 1.8X and 3.8X for the reservoir layer in the datasets HandsOutlines and Henon Map System with negligible performance losses.

B. Hardware Implementation for Compressed RC

Our CRC framework maps the pruned and quantized ESN models to FPGA in the form of a direct logic implementation. All computations are fully unrolled, and weights are hardwired into look-up tables (LUTs) to eliminate memory accesses [16], [17]. Such an approach promises ultra-low latency and extreme- throughput. Since the available logic resources limit the size of the ESN that can be mapped to an FPGA, the approach is targeted towards small and medium sized ESN typically found in edge

applications. Further, compression techniques as discussed in this paper are not only vital to achieve efficient hardware resource usage and reduced energy consumption, but also to improve scalability. Our framework generates ESN designs in Verilog, which are then synthesized with Xilinx Vivado 2022.2 to the Virtex Ultra-Scale xcvu19p-fsvb3824-1-e device.

Table I compares the impact of quantization, pruning, and the combination of them (compressed RC) relative to a baseline RC on the metrics hardware utilization, throughput, latency, and Power Delay Product (PDP). The baseline RC is without any pruning, but also a fully unrolled streamlined design with 32-bit fixed-point quantization, and piecewise linear approximation for the activation function. The other quantized designs use 8- bit quantization and the successive multi-threshold approach to realize the activation function. The reported results for latency and PDP are for running a regression/classification on one input vector.

Table I shows that the baseline RC achieves a high throughput of 120 Msps and a latency of 8.34 ns, at rather high hardware costs. By applying the proposed streamline quantization for the same size network, the resource usage drops significantly. By leveraging quantization and pruning (compressed model), it is possible to further reduce hardware costs down to 12× and 8× in LUTs and FFs, respectively, with maximum throughput and minimum latency. The compressed model also achieves the lowest PDP, with 10× improvement over the baseline RC.

IV. CONCLUSION

We have presented a CRC framework that leverages HSIC LASSO-based pruning and hardware-friendly quantization to compress an RC model for efficient FPGA implementation. The compressed direct logic implementation achieves high throughput and ultra-low latency, up to 188 Megasamples/s and 5.32 ns, respectively, and reduce resource utilization by 12× and energy by 10× compared to a baseline hardware implementation.

ACKNOWLEDGMENT

This work is supported by the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety, and Consumer Protection under grant no. 67KI32004A.

REFERENCES

[1] G. Tanaka et al., “Recent advances in physical reservoir computing: A review,” Neural Networks, vol. 115, pp. 100–123, 2019.

[2] C. Lin et al., “Fpga-based reservoir computing with optimized reservoir node architecture,” in 2022 23rd International Sympo- sium on Quality Electronic Design (ISQED). IEEE, 2022, pp. 1–6.

[3] X. Zhang et al., “Appq-cnn: An adaptive cnns inference accel- erator for synergistically exploiting pruning and quantization based on fpga,” IEEE Transactions on Sustainable Computing, 2024.

[4] H. Wang et al., “Optimizing the echo state network based on mutual information for modeling fed-batch bioprocesses,” Neurocomputing, vol. 225, pp. 111–118, 2017.

[5] D. Li et al., “Structure optimization for echo state network based on contribution,” Tsinghua Science and Technology, vol. 24, no. 1, pp. 97–105, 2018.

[6] J. Huang et al., “Semi-supervised echo state network with partial correlation pruning for time-series variables prediction in industrial processes,” Measurement Science and Technology, vol. 34, no. 9, p. 095106, 2023.

[7] S. Liu et al., “Quantized reservoir computing on edge devices for communication applications,” in 2020 IEEE/ACM Symposium on Edge Computing (SEC), 2020, pp. 445–449.

[8] Y. Abe et al., “Spctre: sparsity-constrained fully-digital reser- voir computing architecture on fpga,” International Journal of Parallel, Emergent and Distributed Systems, vol. 39, no. 2, pp. 197–213, 2024.

[9] A. Jafari et al., “Ultra-low latency and extreme-throughput echo state neural networks on fpga,” in Applied Reconfigurable Computing. Architectures, Tools, and Applications. Springer Nature Switzerland, 2025, pp. 179–195.

[10] N. Trouvain et al., “Reservoirpy: an efficient and user-friendly library to design echo state networks,” in International Confer- ence on Artificial Neural Networks. Springer, 2020, pp. 494– 505.

[11] A. Pappalardo, “Xilinx/brevitas,” 2023. [Online]. Available: https://doi.org/10.5281/zenodo.3333552

[12] Z. Huang et al., “Rethinking the pruning criteria for con- volutional neural network,” Advances in Neural Information Processing Systems, vol. 34, pp. 16 305–16 318, 2021.

[13] H. Ghasemzadeh Mohammadi et al., “Efficient statistical pa- rameter selection for nonlinear modeling of process/performance variation,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 12, pp. 1995–2007, 2016.

[14] M. Yamada et al., “High-dimensional feature selection by feature-wise kernelized lasso,” Neural computation, vol. 26, no. 1, pp. 185–207, 2014.

[15] Y. Umuroglu et al., “Streamlined deployment for quantized neural networks,” https://arxiv.org/abs/1709.04060, 2018.

[16] Logicnets: Co-designed neural networks and circuits for extreme-throughput applications,” in *2020 30th Interna- tional Conference on Field-Programmable Logic and Applica- tions (FPL)*, 2020, pp. 291–297.

[17] A. H. Hadipour *et al.*, “A two-stage approximation methodology for efficient dnn hardware implementation,” in *2025 IEEE 28th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2025, pp. 119–122