2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Evolutionary Multi-objective Quantization of Randomization-based Neural Networks

Javier Del Ser*†, Alain Andrés*, Miren Nekane Bilbao†, Ibai Laña* and Jesus L. Lobo*
*TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
†University of the Basque Country (UPV/EHU), 48013 Bilbao, Bizkaia, Spain
Email: javier.delser@tecnalia.com

***Abstract*—The deployment of Machine Learning models on hardware devices has motivated a notable research activity around different strategies to alleviate their complexity and size. This is the case of neural architecture search or pruning in Deep Learning. This work places its focus on simplifying randomization-based neural networks by discovering fixed-point quantization policies that optimally balance the trade-off between performance and complexity reduction featured by these models. Specifically, we propose a combinatorial formulation of this problem, which we show to be efficiently solvable by multi-objective evolutionary algorithms. A benchmark for time series forecasting with Echo State Networks over 400 datasets reveals that high compression ratios can be achieved at practically admissible levels of performance degradation, showcasing the utility of the proposed problem formulation to deploy reservoir computing models on resource-constrained hardware devices.**

***Index Terms*—Randomization-based neural networks, model quantization, multi-objective optimization, fixed-point arithmetic.**

I. Introduction

Nowadays there is little doubt that Deep neural networks (DNNs) have become the *de facto* machine learning approach for complex tasks, especially modeling spatio-temporal data [1]. Indeed, by stacking several multiple layers of artificial neurons (each comprising simple computations), significant benefits can be brought to a wide range of domains, including computer vision, natural language processing, or speech recognition, among others. Other domains such as finance, biomedicine and transportation have also warmly embraced the unprecedented capability of DNNs to learn complex representations and to provide value in specific problems such as portfolio management, proteomics and traffic modeling.

Unfortunately, DNNs have a large number of parameters that must be learned during training. These models often amount to millions of parameters, which can make them difficult to train and prone to overfitting. At this point, several strategies can be adopted to reduce the complexity of DNNs [2]. One approach is the use of regularization techniques, such as weight-dependent loss penalties or weight decay, which help to prevent overfitting. Another approach is to use simpler architectures, such as smaller networks or shallower layers, which can reduce the number of parameters and make the model more interpretable. Additional techniques such as pruning, quantization, and distillation can be used to further compress the model and make it more efficient at both training and inference times. From a broader perspective, two general strategies can be adopted to reduce the complexity of machine learning models: *ex ante* methods, namely, the reduction of the model complexity during its design, prior to the training process (such as AutoML based on complexity-driven design goals); and *post-hoc* methods, which seek to decrease the complexity of the network after the model has been trained (e.g., pruning).

In this context, an alternative to reduce the complexity is to incorporate randomness in the training process of neural networks. The so-called family of randomization-based neural networks [3] resorts to different forms of randomization to accomplish different goals: 1) complexity reduction (in terms of number of trainable parameters); 2) overfitting avoidance (e.g., dropout, which randomly sets some of the neuron's activations to zero during training); 3) increased robustness against data distribution shifts (e.g., stochastic depth, which randomly removes some of the layers during training to make the model more robust to changes in the data distribution); or 4) estimation of the model's uncertainty (as in Monte Carlo dropout, which uses dropout at test time to estimate the model's uncertainty and make more robust predictions). As a result, randomization-based neural networks have been shown to be effective in improving the performance and robustness of deep learning models, with a growing momentum in computer vision applications, text modeling and time-series classification [4].

In this work we focus on quantization techniques, which are widely used in neural networks to reduce their memory and computational requirements [5]. Quantization involves reducing the precision of the weights and activations in the network, which can be done by mapping them to a fixed set of discrete values. Quantization can be done *ex ante* and *post-hoc*. In *ex ante* quantization, the weights and activations are quantized to lower-precision values, and the network is trained to minimize the loss function in the quantized space. By contrast, in *post-hoc* quantization weights are quantized during inference to make predictions, which can be done with reduced memory and computational overhead. Quantization can also decrease the storage and communication costs when deploying the model to resource-constrained devices.

This work explores the intersection between post-hoc quantization and randomization-based neural networks. We seek the ultimate goal of reducing the computational requirements of this family of neural networks, beyond their small training

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

latencies. To this end, we formulate the discovery of good post-hoc quantization policies as a bi-objective combinatorial optimization problem driven by the trade-off between performance and memory reduction, assuming a trained model. This problem is then efficiently solved by several multi-objective evolutionary algorithms (MOEAs). Experimental results are discussed using one of the most renowned models in this family of neural networks (namely, Echo State Networks, ESNs) and an extensive benchmark of time series forecasting datasets. As revealed by the obtained experimental outcomes, there are clear differences between MOEAs in terms of several quality indicators. Furthermore, we show that memory reductions of more than 80% can be achieved while minimizing the error degradation of the quantized ESN.

The rest of the manuscript is structured as follows: Section II revisits the intersection between evolutionary computation and deep learning, randomization-based neural networks and poses the contribution of this work . Next, Section III formulates the considered optimization problem, and details the MOEAs used to solve it. Next, Section IV states the research questions that experiments aim to address, and discusses on the results obtained for this purpose. Finally, Section V draws conclusions and future research lines.

II. Related work

Before delving into the quantization problem under consideration and the MOEAs used, we briefly revisit the crossroads between evolutionary computation and DNNS (Section II-A) and randomization-based neural networks (Section II-B), ending with an statement on our contribution (Section II-C).

A. Evolutionary computation and neural networks

Evolutionary computation is a class of optimization techniques inspired by the principles of natural evolution (crossover, mutation and survival of the fittest). These techniques can be used to find the optimal values of the parameters of a model, such as neural networks. Evolutionary computation has been used in combination with deep learning to optimize the architecture, hyperparameters, and even the weights of neural networks [6], [7]. Evolutionary algorithms can be used to search for the best architecture of a neural network, by creating a population of architectures and iteratively selecting the best ones based on their performance on a validation set. Examples as such include Evodeep [8], Evo-Unet [9], EvoDCNN [10] or NSGAnet [11], among many others. They can also be used to optimize the hyperparameters of a neural network, such as the learning rate or the dropout rate [12]. These methods can be computationally expensive but have been shown to be effective in finding high-performing architectures, hyperparameters and weights that are not found by traditional grid search or random search methods. Additionally, evolutionary algorithms can be used to optimize the weights of a neural network, by creating a population of weight vectors and iteratively selecting the best ones based on their performance on a validation set. However, there is little consensus whether their good performance shown for training reinforcement learning models can be extrapolated to neural networks of realistic sizes [6]. Evolutionary algorithms have also been used for evolving both binary (*pruning*) and mixed-resolution quantization policies [13]- [17].

B. Randomization-based neural networks

Following up the introduction, randomization-based neural networks overcome issues of conventional neural networks (e.g., computationally demanding training process or overfitting) by embracing randomness during different stages of the model construction process. Among them, we concentrate on those modeling approaches in which part of the trainable parameters are initialized at random, whereas the rest of parameters are learned based on a computationally light training algorithm (e.g. a regularized least squares). This randomization strategy yield learning models with drastically reduced training latencies, making them a suitable choice for constrained computing devices.

Several families of neural networks rely on this strategy. For instance, random vector functional link (RVFL) networks are a type of neural network designed to handle high-dimensional input data [18]. In essence, RVFL is a simple feedforward neural network with an input layer, a hidden layer, and an output layer. The hidden layer uses a random subset of the input features and applies a non-linear activation function to them. The output layer uses the outputs of the hidden layer to make predictions. RVFL networks have been shown to be effective in a variety of tasks such as function approximation, pattern recognition and time-series prediction. When modeling sequential data, ESNs [19] resort to a large random fixed recurrent weight matrix in its hidden layer (a *reservoir*), which endows the model with the capability to learn long-terms dependencies in sequences. The reservoir processes the input data and transfers it to the output layer, which learns the mapping from the hidden layer to the output.

C. Contribution

Randomization-based neural networks have been optimized with evolutionary computation. Evolutionary algorithms can be used to search for the best hyperparameters of an ESN, such as the size of the reservoir, the spectral radius of the recurrent weight matrix, and the regularization coefficients [20], [21]. Additionally, evolutionary algorithms and other metaheuristic approaches can optimize the structure of a randomized neural network, such as the number of neurons, the connectivity patterns, the types of non-linearities or even the values of the weights themselves [22], leading to models with better performance, increased robustness and smaller size.

Scarce works have dealt so far with quantization in randomization-based neural networks. The work in [23] proposes integer-coded reservoir parameters and efficient cyclic shift operations to yield ESN implementations with reduced memory footprint and improved computational efficiency. Likewise, a novel quantization approach suitable for deploying ESNs on edge devices was proposed in [24]. A naive ternary quantization strategy for fixed-point ESN implementations was presented in [25]. Quantization has been also under

study for other members of the randomization-based neural networks, including Extreme Learning Machines [26] and the aforementioned RVFL networks [27], [28].

However, to the best of our knowledge no prior work has explored the application of MOEAs to the post-hoc quantization of randomization-based neural networks. We hypothesize that this exercise can help decide which quantization policy to apply in practical settings based on the estimated Pareto front between modeling performance and memory footprint.

III. Problem statement and proposed solvers

We assume a supervised learning task over a dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathcal{X}$ denotes the n-th instance of the dataset and $y_n \in \mathcal{Y}$ its annotation. We denote as $M_{\boldsymbol{\theta}}$ a model with trainable parameters $\boldsymbol{\theta} = \{\theta_m\}_{m=1}^{M}$ learned from $\mathcal{D}$. For instance, if $M_{\boldsymbol{\theta}}$ is an ESN, $\boldsymbol{\theta}$ should comprise the weights of the input ($\mathbf{W}_{in}$), reservoir ($\mathbf{W}_{res}$) and output (*readout*) matrices of the model, as well as other matrices featured by non-standard formulations of this randomization-based neural network (e.g. feedback matrix $\mathbf{W}_{fb}$). We assume that after training, $\theta_m \in \mathbb{R}\ \forall m \in \{1, \ldots, M\}$. A Q-depth quantization policy is a mapping $f : \{1, \ldots, M\} \mapsto \{1, \ldots, Q\}$ that reduces the bit precision of every learned parameter of the model θ_m. Assuming that training has been produced on a computer with fixed 64-bit precision, the memory reduction Δ_{MEM} achieved by the quantization policy is given by:

$$\Delta_{\text{MEM}}(f) = \sum_{m=1}^{M} f(m)/(64 \cdot M). \quad [\%] \tag{1}$$

Likewise, the generalization performance of the quantized model can be evaluated over a validation dataset $\mathcal{D}_{val} \doteq \{\mathbf{x}_n^{val}, y_n^{val}\}_{n=1}^{N_{val}}$, drawing quantitative measurements that depend on the task being tackled (e.g. Root Mean Square Error for regression problems). By referring to this score as $S(\mathbf{y}^{val}, \widehat{\mathbf{y}}^{val}; \boldsymbol{\theta}, f)$, where $\widehat{\mathbf{y}}^{val}$ denotes the prediction for $\mathbf{y}^{val}$, the bi-objective optimization problem under study is:

$$\mathbf{f}^* = \arg_{f \in \mathcal{F}} \left[\min S(\mathbf{y}^{val}, \widehat{\mathbf{y}}^{val}; \boldsymbol{\theta}, f),\ \min \Delta_{\text{MEM}}(f)\right], \tag{2}$$

where $\mathbf{f}^*$ denotes the set of quantization policies that optimally balance the trade-off between generalization performance and the achieved level of compression of the parameters of the trained model (i.e., the estimated Pareto front between such objectives). This combinatorial formulation essentially comprises a search over the space of possible quantization policies $\mathcal{F}$, whose cardinality $|\mathcal{F}| = Q^M$ grows fast with the number of quantization levels Q.

A. Solution encoding and search operators

The exponentially increasing dimensions of the combinatorial solution space characterizing the problem stated above requires efficient means to perform the search. For this purpose, MOEAs are a natural choice given their renowned capacity to cope with complex combinatorial optimization problems driven by several objectives that conflict with each other. The good performance and design flexibility of MOEAs to include search operators suited to deal with the particularities of the problem at hand have been leveraged over the years in many practical scenarios, including operations research, industry or health, to cite a few [29], [30]. In general, the definition of MOEAs requires specifying:

a) Solution encoding: Since solutions of our problem are quantization policies $f : \{1, \ldots, M\} \mapsto \{1, \ldots, Q\}$, each solution is encoded as an integer vector of size M, where each integer entry can take on values from $\{1, \ldots, Q\}$.

b) Search operators: Search operators are used during the search to evolve a *population* of solutions that represents the knowledge of the solver and ultimately, the Pareto front estimated by the MOEA at hand. In this work two different operators are used. First, the so-called Simulated Binary Crossover (SBX) combines two parent individuals by randomly selecting a crossover point and then using a probability distribution function to determine how the genetic information from the two parents is combined to form the offspring. Second, a neighborhood mutation operator randomly modifies the values of the components of a given solution (based on a probabilistic parameter) to a value drawn uniformly at random from its neighboring values in its alphabet $\{1, \ldots, Q\}$.

The above search operators and encoding strategies can be used by different MOEAs, each comprising distinct strategies to rank solutions and promote convergence and diversity in the estimated Pareto front. The experiments discussed in this work comprise a benchmark between five well-known MOEAs in the literature: Non-dominated Sorting Genetic Algorithm II (NSGA2 [31]), Non-dominated Sorting Genetic Algorithm III (NSGA3 [32]), Strength Pareto Evolutionary Algorithm II (SPEA2 [33]), MultiObjective Cellular genetic algorithm (MOCELL [34]), and Indicator-Based Evolutionary Algorithm (IBEA [35]). We refer to these references for further details about these algorithms, including descriptions of their ranking procedures and diversity preservation mechanisms.

IV. Experiments and results

To shed light on the performance of MOEAs when tackling the bi-objective problem under consideration, an extensive experimental setup has been designed to inform two different research questions (RQs) with empirical evidence:

- RQ1: Are there performance differences between MOEAs when addressing this problem?
- RQ2: Which compression-performance trade-offs can be achieved w.r.t. floating-point or fixed-point single-resolution model implementations?

To this end, we focus on a specific modeling task: time series forecasting with horizons $H \in \{1, 2, 4, 8\}$, meaning that the model aims to predict the value of the time series at $t+H$, with t denoting the time at which the model is queried. This task is formulated over the 400 time series datasets comprised in the Libra benchmark [36]. This benchmark comprises four different use cases, each covering 100 heterogeneous time series from different application domains. To undertake the forecasting tasks over the considered datasets, a single-layer ESN is selected as the model $M_{\boldsymbol{\theta}}$ with parameters $\mathbf{W}_{in}$, $\mathbf{W}_{res}$ and readout (a ridge regressor with regularization

parameter 10^{-5}). For all cases the size of the reservoir is 50 neurons, and train-validation-test partitions are drawn from the time series in a proportion 70-20-10%. The predictive score $S(\boldsymbol{y}^{val}, \widehat{\boldsymbol{y}}^{val}; \boldsymbol{\theta}, f)$ that represents the first objective in Expression (2) is chosen to be the Root Mean Square Error (RMSE) over the validation partition (valRMSE).

In all cases a maximum resolution per weight of $Q = 16$ bits was configured. Population size (50), crossover (0.85) and mutation (0.1) probabilities are set equal for the search operators utilized in all MOEAs, so as to compare them only in terms of their ranking, selection and diversity preservation criteria. The software implementation of these MOEAs available in the jMetalPy package [37] has been used, whereas finite-precision fixed-point arithmetic operations are provided by the Simple Python Fixed-Point Module (SPFPM) Python toolkit[1].

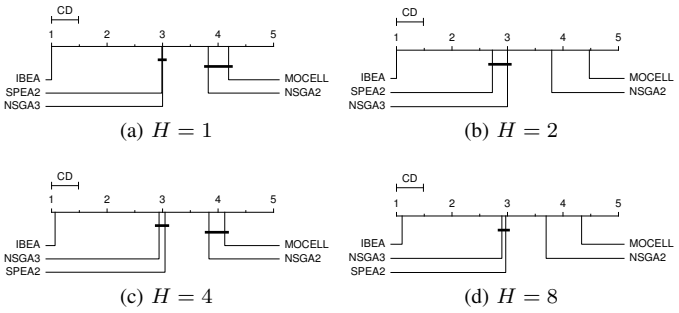
Comparisons are held in terms of several multi-objective quality indicators, namely, normalized HyperVolume (HV), modified Inverted Generational Distance (IGD+) and Epsilon unary indicator (EPS). The global set of non-dominated solutions for all algorithms and runs is considered to be the reference Pareto front for the computation of IGD+ and EPS. For each (MOEA, dataset) combination, 20 independent runs are performed to account for the statistical variability of the results due to the stochastic nature of the search operators. Each run is stopped after 10^4 objective evaluations. Once such runs have been completed, we compute a non-parametric Wilcoxon rank-sum test between the HV values obtained by every pair of MOEAs (20 values per algorithm) for a given dataset to ascertain whether significant differences exist between them. We then use this significance to compute their fractional ranks, granting the same rank to those algorithms for which no significant differences were declared. These ranks are averaged across datasets and graphically represented, so that algorithms whose average ranks are separated by more than a *critical distance* (given by a Nemenyi test with significance level equal to 0.05) can be claimed to perform differently with statistical significance. This ranking procedure is then repeated for different forecasting horizons H.

We now discuss over the results reported for every RQ:

RQ1: Performance differences between MOEAs

Figure 1 depicts the critical distance diagrams computed over the HV results for every forecasting horizon. The length of the line denoted as CD indicates the minimum difference between two average ranks for their corresponding MOEAs to perform statistically different to each other. We first observe that IBEA outperforms clearly its counterparts in this benchmark, followed by SPEA2 and NSGA3 (with no clear winner among them). MOCELL is consistently surpassed by the rest of approaches, except for $H = 4$, where it is found to perform similarly to NSGA2.

A more detailed analysis of these results can be done if we inspect the distribution of the quality indicators for all algorithms. This is the purpose of Figure 2, where the distribution of the HV (first row), IGD+ (second row) and EPS (third row) are shown for all forecasting horizon. The fourth row corresponds to the statistics computed over the total time taken by every solver to complete the search for a given run. It is straightforward to note that the HV results (first row) match the conclusions drawn from Figure 1: a clear dominance of IBEA for all forecasting horizons. However, we observe that IBEA also performs better than the rest of the MOEAs for the EPS indicator. However, there is no consistent winner when it comes to the IGD+ indicator and the running time.



(a) $H = 1$ (b) $H = 2$ (c) $H = 4$ (d) $H = 8$

Fig. 1: Critical distance diagrams depicting the average ranks of the MOEAs under comparison, computed over a dataset of 400 time series and the HV results obtained by these solvers.

Finally, we complement the results for RQ1 by examining the composition of the global Pareto front estimation for some few datasets and forecasting horizons. This analysis allows determining which algorithm discovers results (i.e. quantization policies) that are not dominated by any other algorithm in any of its runs. Figure 3 summarizes this analysis by showing the global Pareto front estimation for different datasets, highlighting with a different color the solver that discovered each of its points. The first objective is normalized by its maximum value to ease the interpretation of the plots. As shown in these particular cases and the statistics of the % of the global Pareto front contributed by each algorithm (boxplots on the right of this figure), IBEA provides a notably higher fraction of the points belonging to the global Pareto estimation, especially for low values of $\Delta_{\text{MEM}}(\cdot)$.

RQ2: Compression-performance trade-offs

Regarding the second research question, Table I shows statistics (mean $\pm$ std) computed over all datasets of the RMSE degradation in validation (first row) and test partitions (second row), as well as the memory reduction Δ_{MEM} achieved by the policies f of the estimated Pareto yielding the minimum value of valRMSE (first column), minimum Δ_{MEM} (second column), and minimum value of the sum of the two objectives. The RMSE degradation Δ_{RMSE} is computed as the normalized difference between the RMSE achieved by the model quantized by the selected policy and the RMSE obtained with a fixed-point quantization of $Q = 16$ bits per parameter: the lower Δ_{RMSE} is, the lower the RMSE degradation of the quantized model will be compared to its non-quantized version.

[1]Simple Python Fixed-Point Module (SPFPM), https://github.com/rwpenney/spfpm, accessed on July 9th, 2023.

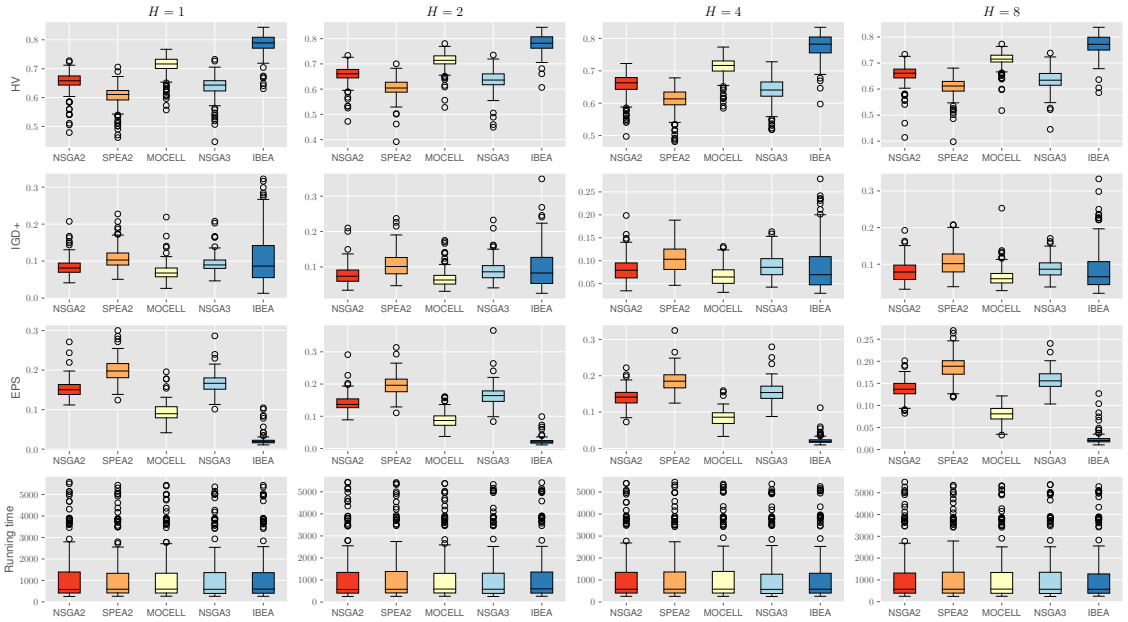


Fig. 2: Boxplots of the multiobjective quality indicators computed over all datasets for different forecasting horizons (columns).

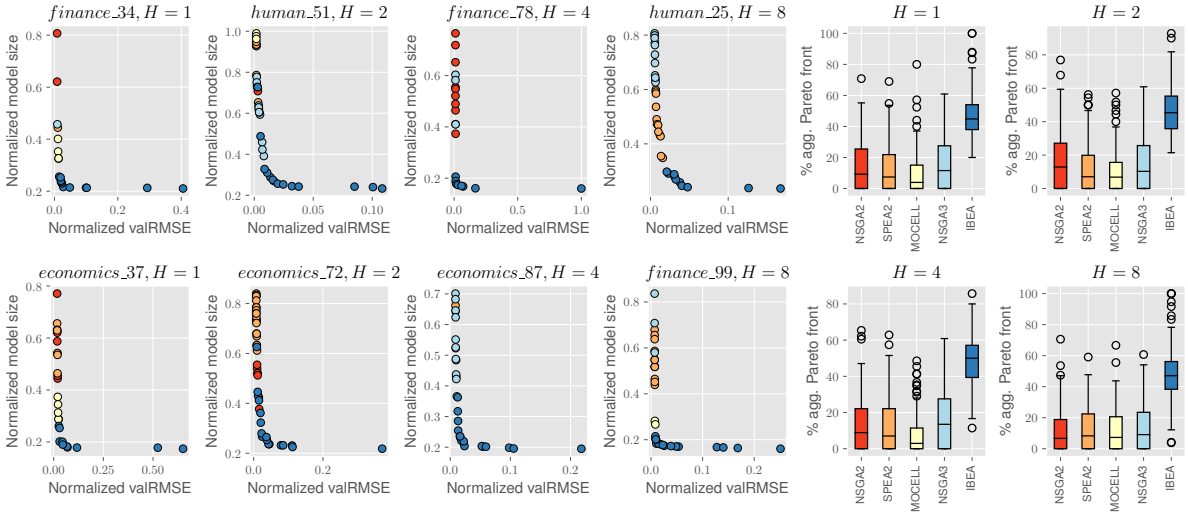


Fig. 3: Contribution of each solver to the global Pareto front of some datasets (left) and distribution over all datasets (right).

Several observations can be done: to begin with, compression levels between 18% and 54% can be achieved at an RMSE degradation whose tolerance will depend on the application at hand. Above all, we highlight the results shaded in gray in the table, which expose that for long forecasting horizons (high H), the quantization policy yields lower RMSE values than the case where the resolution of all model parameters is kept to $Q = 16$. This can be also noted in the decreasing behavior of Δ_{RMSE} with H for any selected policy. This counter-intuitive result can be due to the fact that quantization can be regarded as a post-tuning process of the ESN parameters which, as the forecasting horizon increases, allows the compressed model to fit better the dynamics of the time series under consideration. This, however, requires further analysis in the near future.

V. Conclusions and Future Work

This manuscript has showcased the benefits of MOEAs for efficiently delineating the Pareto relationship between performance and model compression ratio that can be achieved by quantizing randomization-based neural networks. Focusing on one of these models (ESN) and time series forecasting tasks, we have shown throughout extensive experimentation that the memory footprint of trained ESNs can be dramatically reduced at a penalty in the predictive performance of the quantized model. We have also unveiled an interesting parameter refining effect of post-hoc quantization by which, in addition to model

TABLE I: RMSE degradation and model compression ratio statistics of selected policies belonging to the Pareto front estimated for each dataset.

	H	f @ min valRMSE	f @ min Δ_{MEM}	f @ min(valRMSE + Δ_{MEM})
Δ_{RMSE} (val, %)	1	50.65 ± 29.62	4250.03 ± 649.63	369.75 ± 115.02
	2	10.72 ± 9.11	4078.38 ± 541.14	272.64 ± 45.43
	4	-9.42 ± 3.89	4278.84 ± 617.49	211.95 ± 32.31
	8	-15.42 ± 3.34	3751.14 ± 513.07	224.64 ± 39.35
Δ_{RMSE} (test, %)	1	63.95 ± 29.73	3937.43 ± 606.45	375.60 ± 123.88
	2	19.87 ± 16.56	3965.52 ± 555.58	297.68 ± 64.01
	4	-2.55 ± 5.54	4120.02 ± 668.07	208.49 ± 36.91
	8	-4.63 ± 5.28	3311.40 ± 480.80	193.65 ± 31.67
Δ_{MEM} (%)	1	51.37 ± 25.81	18.01 ± 2.16	31.12 ± 14.82
	2	54.03 ± 26.53	18.60 ± 2.69	35.03 ± 16.34
	4	54.21 ± 25.85	18.92 ± 2.78	36.51 ± 15.76
	8	53.65 ± 27.18	19.18 ± 3.14	35.63 ± 16.98

compression, fine-tuning can be achieved, leading to improved generalization performance for high forecasting horizons.

Besides investigating this effect, future research will extend the problem to architecture search, leveraging the modularity and design flexibility of reservoir computing models. Specialized readout models will be also studied, featuring learning algorithms that are sensitive to fixed-point arithmetic, which can support the *ex ante* optimization of quantization policies.

ACKNOWLEDGMENTS

The authors thank the Basque Government (MATHMODE, ref. T1256-22, and BEREZ-IA, ref. KK-2023/00012) and *Euskampus Fundazioa* (ref. ORLEG-IA) for their support.

REFERENCES

[1] S. Wang, J. Cao, and S. Y. Philip, "Deep learning for spatio-temporal data mining: A survey," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 8, pp. 3681–3700, 2020.

[2] X. Hu *et al.*, "Model complexity of deep learning: A survey," *Knowledge and Information Systems*, vol. 63, pp. 2585–2619, 2021.

[3] L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," *Information Sciences*, vol. 364, pp. 146–155, 2016.

[4] S. Scardapane and D. Wang, "Randomness in neural networks: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 2, p. e1200, 2017.

[5] Y. Guo, "A survey on methods and theories of quantized neural networks," *arXiv preprint arXiv:1808.04752*, 2018.

[6] A. D. Martinez *et al.*, "Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges."

[7] H. T. Ünal and F. Başçiftçi, "Evolutionary design of neural network architectures: a review of three decades of research," *Artificial Intelligence Review*, pp. 1–80, 2022.

[8] A. Martín *et al.*, "Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation."

[9] T. Hassanzadeh, D. Essam, and R. Sarker, "Evou-net: an evolutionary deep fully convolutional neural network for medical image segmentation," in *Annual ACM Symposium on Applied Computing*, 2020, pp. 181–189.

[10] ——, "EvoDCNN: An evolutionary deep convolutional neural network for image classification," *Neurocomputing*, vol. 488, pp. 271–283, 2022.

[11] Z. Lu *et al.*, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[12] P. Jiang, Y. Xue, and F. Neri, "Continuously evolving dropout with multi-objective evolutionary optimisation," *Engineering Applications of Artificial Intelligence*, vol. 124, p. 106504, 2023.

[13] Z. Wang *et al.*, "Evolutionary multi-objective model compression for deep neural networks," *IEEE Computational Intelligence Magazine*, vol. 16, no. 3, pp. 10–21, 2021.

[14] T. Wang *et al.*, "Apq: Joint search for network architecture, pruning and quantization policy," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2078–2087.

[15] Z. Liu *et al.*, "Evolutionary quantization of neural networks with mixed-precision," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 2785–2789.

[16] Y. Yuan *et al.*, "Evoq: Mixed precision quantization of DNNs via sensitivity guided evolutionary search," in *International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.

[17] Z. Wang *et al.*, "Adaptive integer quantisation for convolutional neural networks through evolutionary algorithms," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021, pp. 1–7.

[18] A. K. Malik *et al.*, "Random vector functional link network: recent developments, applications, and future directions," *Applied Soft Computing*, p. 110377, 2023.

[19] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks - with an erratum note," *German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[20] L. Wang *et al.*, "Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm," *Energy*, vol. 153, pp. 801–815, 2018.

[21] H. Wang and X. Yan, "Optimizing the echo state network with a binary particle swarm optimization algorithm," *Knowledge-Based Systems*, vol. 86, pp. 182–193, 2015.

[22] N. Chouikhi *et al.*, "PSO-based analysis of echo state network parameters for time series forecasting," *Applied Soft Computing*, vol. 55, pp. 211–225, 2017.

[23] D. Kleyko *et al.*, "Integer echo state networks: efficient reservoir computing for digital hardware," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1688–1701, 2020.

[24] S. Liu, L. Liu, and Y. Yi, "Quantized reservoir computing on edge devices for communication applications," in *IEEE/ACM Symposium on Edge Computing (SEC)*, 2020, pp. 445–449.

[25] K. Honda and H. Tamukoh, "A hardware-oriented echo state network and its FPGA implementation," *Journal of Robotics, Networking and Artificial Life*, vol. 7, no. 1, pp. 58–62, 2020.

[26] H. H. Bosman *et al.*, "Online extreme learning on fixed-point sensor networks," in *IEEE International Conference on Data Mining Workshops*, 2013, pp. 319–326.

[27] A. Rosato, R. Altilio, and M. Panella, "On-line learning of RVFL neural networks on finite precision hardware," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[28] D. Kleyko *et al.*, "Density encoding enables resource-efficient randomly connected neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3777–3783, 2020.

[29] T. Stewart *et al.*, "Real-world applications of multiobjective optimization," *Multiobjective optimization: interactive and evolutionary approaches*, pp. 285–327, 2008.

[30] C. A. Coello Coello, "Multi-objective evolutionary algorithms in real-world applications: Some recent results and current challenges," *Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences*, pp. 3–18, 2015.

[31] K. Deb *et al.*, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[32] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 577–601, 2014.

[33] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," *TIK report*, vol. 103, 2001.

[34] A. J. Nebro *et al.*, "MOCELL: A cellular genetic algorithm for multiobjective optimization," *International Journal of Intelligent Systems*, vol. 24, no. 7, pp. 726–746, 2009.

[35] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *International conference on parallel problem solving from nature*. Springer, 2004, pp. 832–842.

[36] A. Bauer *et al.*, "Libra: A Benchmark for Time Series Forecasting Methods," in *ACM/SPEC International Conference on Performance Engineering*, 2021.

[37] A. Benítez-Hidalgo *et al.*, "jMetalPy: A Python framework for multi-objective optimization with metaheuristics," *Swarm and Evolutionary Computation*, vol. 51, p. 100598, 2019.